LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

PERSONAL SOFTWARE

ANNO 1 N. 3
DICEMBRE 1982 L. 3.500

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

Spedizione in abb. postale gruppo III/70

GENERAZIONE DI LABIRINTI

COMUNICAZIONI TRA PET E PET

PROGRAMMI COMPLETI PER APPLE - PET/CBM TRS/80 - SINCLAIR ATOM - VIC 20 - ATARI

RICERCA SU ALBERI

COMPRESSIONE DI TESTI

HARDEN

ha scelto per Voi

Il minicomputer al prezzo di un personal.
memoria 128 Kbytes espandibile a 896 KBytes.
dischi 1.2 Mbytes espandibile a 10 Mbytes.
Microprocessore Intel 8088'® a 16 bits.
Sistemi operativi: CP/M86®, MS DOS®
Linguaggi: BASIC, CBASIC, Assembler, COBOL, Pascal, Fortran...
Il Sirius 1 il numero 1 della nuova generazione dei personal computers.

Harden-Sirius, un binomio che non teme confronti.
Sirius Systems Technology Inc.:
l'hardware superbo,
il software di base all'avanguardia
Harden S.p.A.:
l'organizzazione,
la serietà,
la competenza
La certezza di un giusto acquisto.

HARDEN S.p.a. - 26048 SOSPIRO (CR) Italia - Tel. 0372/63136 r.a. - Telex 320588 I

ELEDRA 3S spa - Viale Elvezia, 18 - 20154 Milano

PUTER
GIUGNO 1982
1

RICHIESTA DI ABBONAMENTO GRATUITO

Spedire il coupon in busta chiusa a:
ELEDRA 3S S.p.A., - Viale Elvezia, 18 - 20154 Milano PS 3

☐ Desidero ricevere regolarmente Eledra Personal Computer News
☐ Ricevo già EPCN. Desidero avere informazioni sul Jolly
☐ Indicatemi il vostro rivenditore più vicino

Cognome e nome ____________________

Tit. __________ Attività ____________________

Indirizzo ____________________

CAP ________ Città ____________________ Tel. ____ / ________

PERSONAL SOFTWARE

ANNO 1 N. 3 DICEMBRE 1982

Sommario

RUBRICHE

ARTICOLI

PROGRAMMI

IL BASIC

PROGRAMMI PRATICI IN BASIC
di Lon POOLE

Il libro è una raccolta di programmi di tipo finanziario, matematico, scientifico e di decisioni manageriali. Ogni programma, orientato alla risoluzione di un problema pratico, è presentato con una breve descrizione iniziale, un campione di esecuzione, il listing BASIC, nonchè, per molti, una sezione in cui sono raccolte possibili variazioni per rendere il programma stesso più rispondente alle necessità personali. I programmi sono stati scritti in un BASIC generale, il che li rende, per la maggior parte, direttamente utilizzabili, senza alcun cambiamento, su molti microcomputer, e sono stati provati usando varie versioni di BASIC.

SOMMARIO

Reddito medio - Valore corrente di un buono del tesoro - Calcolo dell'interesse di obbligazioni - Interesse continuo composto - Regola dell'interesse 78 - Valore netto presente di un investimento - Flusso di cassa non uniforme - Affitto/decisione di acquisto - Analisi degli investimenti sindacali - Scambio di deprezzamento - Ripartizione di quote - Quota interna di ritorno - Amministrazione finanziaria - Analisi di quote di stato finanziario - Partecipazione ai profitti dei contribuenti - Controllo dei libri - Bilancio di casa - Metodo critico Path (CPM) - Pert - Algoritmo di trasporto - Teoria delle code - Analisi di Markov - Analisi non lineare di Breakeven - Analisi con la metrice dei vantaggi - Decisione di Bayes - Quantità economica di un ordine - Quantità economica di una produzione - Teoria della stima statistica.

cod. 550D pag. 200 L. 12.500

INTRODUZIONE AL BASIC

Si tratta di un vero e proprio corso di BASIC. Le caratteristiche che lo hanno fatto scegliere, per questi mini elaboratori sono di essere facile da apprendere ed utilizzare nonchè di essere un linguaggio interattivo. Se ci sono errori, questi possono subito essere rilevati in maniera tale da poterli correggere.
Facile da leggere e imparare, che con numerosi esempi "testa" subito il reale apprendimento raggiunto dal lettore.
Un testo che si rivolge ai principianti. Infatti in maniera progressiva e pedagogica, senza alcuna necessità di formazione di base sulle tecniche di informatica, illustra, spiega, esemplifica tutti gli aspetti dei linguaggi attualmente disponibili su differenti sistemi, che vanno dal microcalcolatore ai sistemi time-sharing chi ha già acquisito esperienza in altri linguaggi, invece potrà saltare la parte preliminare, di introduzione alla materia, per entrare subito nel vivo del BASIC.
La base dell'informatica; le generalità del linguaggio BASIC; le istruzioni; il trattamento degli elenchi; tabelle, file, sottoprogrammi; i procedimenti grafici e le possibilità offerte; le istruzioni specifiche di alcuni sistemi.

cod. 502A pag. 324 L. 18.500

PROGRAMMARE IN BASIC
di Michel PLOUIN

Come tutte "le lingue viventi", il BASIC viene applicato in realtà a questa o a quella macchina sotto forma di dialetti più o meno particolari. Questo libro si sforza di descrivere in modo metodico il BASIC delle tre macchine più diffuse sul mercato mondiale: Apple, PET, TRS 80, e, naturalmente, i loro derivati.
Ciò faciliterà anche la conversione di programmi scritti, da un determinato personal computer agli altri. Numerosi esempi (programmi verificati attentamente) chiariscono i concetti proposti e sono immediatamente riutilizzabili da i possessori dei sopracitati personal.

SOMMARIO

Introduzione - Le variabili - Funzioni - Logica di svolgimento di un programma - Dialogo con la macchina - Funzioni speciali - Effetti grafici ed altri - Preparazione dei programmi codice ASCII e caratteri speciali - Calcolo binario ed esadecimale - Esempi di programmi.

cod. 513A pag. 94 L. 8.000

COME PROGRAMMARE
di Jean Claude BARBANCE

Il libro insegna a chi programma come deve enunciare e definire correttamente l'idea iniziale, come analizzarla e trasformarla, e come verificare la correttezza della stessa sino a giungere alla stesura di un programma ben documentato, leggibile e facilmente modificabile. Vengono esplicitate tutte le altre fasi intermedie del lavoro: le vie alternative che si presentano e tra cui scegliere, le eventuali estensioni, le prove e le verifiche che occorre fare per ottenere un programma conforme a quanto ci si era proposti. Poichè era necessario appoggiarsi a un linguaggio, si è scelto il BASIC per la sua larga diffusione. I concetti esposti, comunque sono utilizzabili con qualsiasi altro linguaggio. I programmi presentati sono stati tutti provati e girano su computer da 4 a 64K di memoria.

SOMMARIO

Realizzazione dei programmi: le fasi - La definizione degli obiettivi - L'analisi - La codifica e la messa a punto del programma - Presentazione degli esempi - Rappresentazione di un numero decimale mediante una stringa di caratteri alfabetici - Il gioco del 421 - La contabilità personale.

cod. 511A pag. 192 L. 12.000

SCONTO 20% agli abbonati fino al 28-2-83

GRUPPO EDITORIALE JACKSON
Divisione Libri

Per ordinare il volume utilizzare l'apposito tagliando inserito in fondo alla rivista.

Editoriale

Le novità di questo mese

Mauro Boscarol

Molte novità in questo numero. Personalmente, ritengo che la più interessante sia la nuova rubrica *I segreti dei personal*. Si tratta di un grosso spazio, il cui tema generale è appunto "i segreti del personal", cioè i trucchi, gli accorgimenti, per usare più a fondo il vostro computer, e le funzioni nascoste, le caratteristiche più "intime" delle vostre macchine, quello che il manuale non dice. Lo spazio dei "segreti" è suddiviso in sottorubriche: in questo numero ci sono quelle del PET/CBM, TRS-80 mod. I, VIC, ZX81. Nei prossimi numeri ancora questi e molti di più. Dipende anche da voi.

Un'altra novità: la posta. Tutte le riviste hanno una rubrica di posta, noi non potevamo farne a meno. Ma la nostra posta non è concentrata nella rubrica: è anche sparsa per il giornale. Se ci scrivete indicando un miglioramento per una routine pubblicata nella *Raccolta di routine Basic* troverete là il vostro suggerimento. Se ci scrivete per parlare di qualche "gioco informatico", è in questa rubrica che troverete le vostre richieste o i vostri suggerimenti.

Un'ultima novità è il servizio programma: per i programmi più lunghi e complessi, questo servizio vi permette di richiedere un disco già registrato direttamente a casa vostra, pagando in contrassegno. Per maggiori dettagli consultate la sezione programmi.

Per il resto, nulla di nuovo, ossia articoli e programmi. Ma articoli interessanti, come *Comunicazione da PET a PET attraverso la "User Port"* che vi insegna come collegare due PET tra di loro in modo che si scambino informazioni. E poi *La compressione dei testi* tratta il problema sempre attuale di ridurre l'occupazione su disco dei file. *Ricerca su alberi*, in questa seconda parte affronta le tecniche di tipo euristico. *Generatore di labirinti* descrive un algoritmo che produce labirinti casuali di ogni formato direttamente sul video.

E programmi interessanti. *La carta del cielo* è il presupposto scientifico dell'astrologia. Mi spiego meglio. La redazione di un oroscopo avviene in due passi. Dapprima si determina la posizione dei pianeti, del sole e della luna al momento della nascita dell'individuo. Questa prima fase, cioè il tracciamento della "carta del cielo" in una certa data, si realizza mediante le tecniche dell'astronomia, e quindi la sua scientificità è fuori dubbio. Si tratta di calcoli complicati ma ben determinati: li potere eseguire da soli, e con l'esattezza dei minuti secondi, con il programma riportato in fondo alla rivista. Il secondo passò, l'interpretazione della carta del cielo, è a questo punto lasciata a voi: non si tratta, fino ad oggi almeno, di certezza scientifica, ed ognuno ha le sue tecniche personali.

Per il TRS-80 c'è poi il *Backgammon* scritto da Adam Scott, l'inventore delle *Adventure*, i programmi che vi pongono al centro di una azione di cui siete protagonista (sono molto difficili da tradurre ma ci stiamo provando). Qui Adam Scott ha temporaneamente abbandonato le *Adventure* per scrivere questo Backgammon, con il quale potete sfidare il vostro calcolatore.

E altri programmi per VIC, Atari, ZX81, Atom.

Buona lettura.

Nel prossimo numero:

Tecniche di ricerca

Grafica tridimensionale con l'Apple

Interi in precisione multipla

Programmi per VIC, ZX81, Atari, PET/CBM, Apple

La programmazione dei giochi di movimento

Enrico Ferreguti

I programmi di giochi, si sa, sono quelli che più vengono usati dagli utenti di home computer e più specificatamente dello ZX 81. È mia volontà non toccare l'argomento del linguaggio macchina poiché credo che gli utilizzatori di queste macchine, almeno la maggior parte, non lo conoscano molto bene. Lascio fuori anche i possessori di ZX 80 invitandoli a leggermi ugualmente (diverse case di hardware commercializzano aggeggi in grado di attivare lo slow anche su ZX 80 nuova ROM, basta cercare un po'...).

Il principio generale della programmazione di giochi di movimento si basa sul cancellare la figura da spostare e ridisegnarla in un'altra posizione, calcolata secondo parametri influenzati dall'utente.

Un gioco, per essere tale, deve essere interattivo, cioè deve stabilirsi un legame fisico tra giocatore e calcolatore. Questo deve avvenire tramite l'unico organo di input di cui è in possesso il Sinclair: la tastiera.

Se nei giochi tradizionali è lasciata la possibilità di pensare fermando il calcolatore con degli input, questo non è possibile con un gioco di movimento i cui principali ingredienti sono prontezza di riflessi ed intuizione. Eliminata l'INPUT l'istruzione che fa per noi è la funzione INKEY$, che legge da programma il tasto premuto sulla tastiera in quel momento, senza fermarsi. L'uso è molto semplice, p.es.:

```
10 IF INKEY$"" THEN GOTO 10
20 LET A$=INKEY$
30 PRINT "HAI PREMUTO IL TASTO"; A$
```

Questo semplice programma stampa sul video il carattere premuto. La linea 10 si ripete finché non si preme un tasto (finché INKEY$ sarà diverso dalla stringa nulla); la linea 20 legge il tasto premuto (che sarà anche quello che ha fatto uscire lo ZX dal ciclo in 10) ed il gioco è fatto.

La seconda considerazione che si deve fare è nell'uso di CLS; mentre nei programmi per la versione non espansa l'uso di CLS sarà frequentissimo, poiché permette di calcellare la figura e di riscriverla velocemente in un'altra posizione, nei programmi da 16 K il suo uso rischierà di essere un pericoloso ostacolo alla stesura di programmi veloci di grafica animata. La causa è presto spiegata: lo schermo vuoto nello ZX 81 1 K è formato da 24 caratteri NEWLINE in quanto non è mappato (si gonfia o si rimpicciolisce a seconda di quello che stampiamo), quindi il calcolatore quando incontra quest'istruzione si limita a contrarre lo schermo e a porre 24 NEWLINE e quindi l'operazione si rivela sufficientemente veloce. Con l'espansione connessa invece, lo schermo è mappato ed occupa permanentemente 792 byte. L'operazione da compiere diviene inaccettabile in *slow mode* perché il calcolatore è costretto a percorrere tutto il display file ponendo ad ogni locazione (792!) uno spazio bianco.

Passiamo ora ad analizzare il programma 1.

```
  5 SGN
 10 LET H=10
 11 LET S=0
 12 LET A=INT (RND*22)
 15 LET A$="[illegible]"
 17 LET C=INT (RND*15)
 20 IF INKEY$="1" THEN LET H=H-
(H>0)
 25 IF INKEY$="Q" THEN LET H=H+
(H<21)
 26 IF INKEY$="0" THEN PRINT AT
 H,4;"*  -  --"
 27 IF INKEY$="0" AND H=A THEN
GOTO 200
 28 LET A=A+INT (RND*3)-1
 29 LET A=A+(A<0)-(A>15)
 30 CLS
 40 PRINT AT H,0;A$;AT A,C+10;"
<"
 50 LET C=C-1
 60 IF C+10<3 AND A=H THEN GOTO
 120
 65 IF C+10=0 THEN GOTO 250
 70 GOTO 20
120 CLS
122 PRINT AT H,0;")X▆X("
124 PRINT AT 0,0;"FINE GIOCO ";
S
125 PAUSE 200
130 RUN
200 LET S=S+C+10
210 PRINT AT A,C+10;")X("
220 GOTO 12
250 LET S=S-10
255 IF S<-19 THEN GOTO 120
260 GOTO 12
```

Programma 1. *Defender*

Siamo alla guida di un'astronave e non dobbiamo lasciar passare nessun alieno (rappresentato da un ">"). Ogni volta che ne lasciamo passare uno il nostro punteggio subisce un decremento di 20 punti (se arriva a −20 il gioco finisce). Possiamo dividere il programma in 4 diverse sezioni.

1. la sezione di guida da 20 a 29:
   Vediamo che il programma decrementa (fa andare su l'astronave) la variabile H (altezza dell'astronave) nel caso sia premuto il tasto 1 e la incrementa quando si preme Q. Si nota inoltre un esempio di logica relazionale che costringe l'astronave nella fascia da 0 a 21: LET H=H−(H 20), LET H=H+(H 21). Se si preme lo 0 prima si ha la stampa della raffica e in riga 27 si ha il salto alla subroutine che aggiornerà il punteggio nel caso sia premuto 0 e che l'astronave sia sulla stessa linea dell'avversario. Si procede poi nelle righe 28 e 29 al movimento in su o in giù in modo RND controllato dai soliti operatori relazionali (riflettete un attimo sulla riga 29 e cercate di capire, ogni vostra deduzione sarà senz'altro più chiara di ogni mia spiegazione).
2. sezione di stampa (righe 30, 40)
   In riga 30 c'è il CLS che cancella le figure dell'alieno e dell'astronave che vengono ristampate secondo i nuovi parametri.
3. sezione di controllo (righe 50, 65)
   Serve a comandare l'avanzata dell'alieno e a vedere se l'alieno ha colpito l'astonave.
4. sezione di utilità
   Comprende la parte di inizializzazione (5, 17) e le varie routine di asservimento: quella per aggiornare il punteggio e per distruggere l'alieno e quella di fine gioco.

Il programma gira su 1 K di RAM ed è sconsigliabile usarlo con l'espansione da 16 K; per l'uso con quest'ultima bisognerà sostituire il CLS con opportuni mascheramenti di spazi. Vediamo subito la tecnica di visualizzazione con l'espansione da 16 K.

Se volessimo disegnare una figura più complessa di quella vista prima con "Defender", il programma necessariamente rallenterebbe. È per questo che si ricorre alla funzione TAB che si rivela in questi casi molto versatile. Queste 3 linee di programma fanno correre una freccia lungo lo schermo:

```
100 FOR J=0 TO 27
110 PRINT AT 15,J; "□□□▗□"; TAB J;
    "▄▄▄▄▙"; TAB J; "□□□▗▘"
120 NEXT J
```

Il disegno è difficile da riconoscere, ma questo diagramma potrà aiutare.

```
PRINT AT 15, J □□□▗□
TAB J          ▄▄▄▄▙
TAB J          □□□▗▘
```

Possiamo individuare 2 importanti punti:

1. La tecnica dell'uso di TAB per stampare immediatamente sotto l'ultimo disegno, alla posizione J.
2. L'uso di spazi all'inizio dei tre gruppi di caratteri che formano la freccia; questi cancellano immediatamente i resti della precedente freccia quando viene mosso il disegno lungo lo schermo.

```
   5 DIM T(5)
  10 RAND
  20 PRINT "ZX    GARA DELLE TART
ARUGHE    ZX"
  35 LET A=INT (RND*5+1)
  40 LET T(A)=T(A)+1
  45 PRINT AT A*3+4,T(A);"       "
;TAB T(A);" ";CHR$ (A+156);"
";TAB T(A);"  "
  50 IF T(A)<>26 THEN GOTO 35
  60 PRINT AT 5,4;"VINCE LA TART
ARUGA N. ";A
  70 PAUSE 500
  75 CLS
  80 RUN
```

```
ZX    GARA DELLE TARTARUGHE    ZX


     VINCE LA TARTARUGA N. 5
```

Programma 2. *Gara delle tartarughe*

Esaminiamo ora il programma 2 che riprende la tecnica vista prima e ne introduce un'altra. Il programma muove 5 tartarughe lungo lo schermo in modo casuale (1 K).

Tralasciamo la parte di gestione e soffermiamoci sulla riga 45 dove sono concentrati tutti i trucchi. Innanzitutto notiamo le TAB con cui si disegna la figura, poi si nota la funzione CHR$ per stampare il numero della tartaruga in inverso. Al di là del semplice significato grafico, questo trucchetto permette di stampare sullo schermo di una cifra a velocità molto elevata.

Forse ora vorrete provare a fare delle altre animazioni, forse l'articolo vi ha anche suggerito qualche idea; intanto provate il programma 3, scoprite cosa fa e giocate (è da 1 K ed è facilissimo).

```
   5 LET S=NOT PI
  10 LET D=0
  11 LET A=VAL "2"
  12 LET P=10
  15 LET T=INT (RND*20)
  20 LET E=INT (RND*20)
  25 LET K=RND-.5
  27 IF K<NOT PI THEN LET B$="<"
  26 IF K>NOT PI THEN LET B$=">"
  30 LET A$=" X"
  40 CLS
  50 PRINT AT D,INT E;A$;AT 21,T
;"▀"
  55 PRINT AT NOT PI,NOT PI;B$
  57 IF A=SGN PI AND D=20 AND IN
T E+SGN PI=T THEN GOTO 300
  60 IF INKEY$="0" THEN GOSUB 20
0
  70 IF INKEY$="1" AND A=SGN PI
THEN LET E=E-(E>NOT PI)
  80 IF INKEY$="3" AND A=SGN PI
THEN LET E=E+(E<20)
  90 IF D=20 THEN GOTO 500
  92 LET D=D+A
  93 LET E=E+K
  95 LET S=S+P
 100 GOTO 40
 200 LET A$="""X"""
 205 LET P=1
 210 LET A=SGN PI
 250 RETURN
 310 PRINT AT NOT PI,NOT PI;"BEN
ATTERRATO";AT SGN PI,NOT PI;S
 320 PAUSE 100
 330 GOTO VAL "10"
 500 PRINT AT NOT PI,NOT PI;"TI
SEI SCHIANTATO";AT SGN PI,NOT PI
;S
 510 PAUSE 1E3
 520 RUN
```

Programma 3.

Tandy Radio Shack

Per chi ha il modello 1

Renzo Gabrielli

Qualche volta può essere necessario usare la stampante al posto del video per visualizzare dati o altro, che altrimenti andrebbero persi spegnendo il computer. Per esempio, se volete avere una copia di dimostrazione del funzionamento di un programma, dovreste cambiare tutte le istruzioni PRINT in LPRINT. Ecco il consiglio che fornisce la casa costruttrice del TRS-80.

Per trasformare tutte le istruzioni PRINT in LPRINT, battete:

POKE 16414, 141
POKE 16415,5

Per riconvertire le istruzioni, scrivete:

POKE 16414,88
POKE 16415,4

Le istruzioni precedenti possono comparire nel vostro programma, ma possono essere impartite anche da tastiera. Un'altra applicazione: può servire per far girare programmi pronti che prevedono l'uso delle stampante, quando questa manca. Le prossime istruzioni trasformano tutte le LPRINT in PRINT:

POKE 16422,88
POKE 16423,4

Per invertire il procedimento, usate:

POKE 16422,141
POKE 16423,5

Il programma non viene modificato, ma il computer interpreterà le istruzioni in modo differente a seconda dei valori presenti in queste quattro locazioni di memoria.

SHOP
bit
primavera

Hard-copy del video su stampante

Il programma che segue proviene dai tecnici della Tandy Radio Shack:

```
1000 C=15360: D=C+63: FOR A=1 T0 15: FOR
     B=C TO D
1010 E=PEEK(B): LPRINT CHR$(E);
1020 NEXT B
1030 C=C+64: D=D+64
1040 LPRINT
1050 NEXT A
```

Questo programma fa strane cose quando si tenta di stampare caratteri grafici. Il suo uso principale è trasferire su carta il testo che appare sul video.

Come cambiare pagina con LPRINT CHR$(12)

Recentemente ho perso tre ore per cercare di capire perché uno dei miei programmi non cambiava pagina correttamente usando l'istruzione LPRINT CHR$(12). La soluzione era abbastanza semplice e può darsi che qualche lettore possa trarre beneficio da questa mia perdita di tempo.

Come sapete, la locazione 16424 contiene il massimo numero di righe per pagina, mentre la locazione 16425 contiene il numero di righe già stampato. Quando viene mandato un CHR$(12) alla stampante, con una sottrazione viene ottenuto il numero corretto di line feed da mandare alla stampante per posizionarsi all'inizio della nuova pagina. Errore! Queste due locazioni di memoria contengono il numero di LPRINT per ogni pagina e il numero di LPRINT già mandati.

Per esempio sulla mia stampante uso degli LPRINT per modificare il formato di stampa. La stampante non scrive niente, ma la locazione 16425 viene incrementata di uno per ogni cambiamento effettuato. Se dovete modificare il tipo di carattere o la spaziatura o altro da programma, vi sarà capitato lo stesso problema.

Un'altra cosa è che se le due locazioni menzionate prima contengono lo stesso valore, per esempio 67, un'istruzione LPRINT CHR$(12) mandata alla stampante provocherà il blocco del computer mentre la stampante continuerà a sfornare carta. Il solo modo per rimediare è premere RESET e ricominciare da capo.

Doppio schermo sul VIC

Jim Butterfield

Una delle caratteristiche più interessanti del VIC è il controllo sulle possibilità dello schermo. In questo articolo ci occuperemo di far muovere lo schermo. Naturalmente lo schermo in se stesso non si muove, ma rimane dove avete piazzato il vostro televisore o il vostro monitor. Quello che vogliamo fare è cambiare le locazioni di memoria da cui lo schermo prende le sue informazioni.

La schermata

Le informazioni che appaiono sullo schermo provengono dalla memoria del VIC. Ogni carattere sullo schermo corrisponde a un valore in memoria. Ogni locazione sullo schermo è collegata a una specifica locazione di memoria. Ecco il trucco che intendiamo usare: vogliamo cambiare il collegamento, in modo che ogni locazione sullo schermo provenga da una differente cella di memoria. Se riusciamo a fare ciò, potremo memorizzare due schermi completamente separati. Poi, potremo passare alternativamente da uno schermo all'altro per ottenere effetti speciali o un utile doppio schermo.

Nella memoria del VIC esiste una locazione che controlla dove lo schermo è agganciato alla memoria. Possiamo facilmente cambiare il contenuto di questa locazione provocando la comparsa del nuovo schermo. Ma questo non basta.

Lavorare sullo schermo

Dobbiamo cambiare anche i puntatori di lavoro del VIC, quelli che mettono i nuovi caratteri sul video. Non ci sarà di molto aiuto aver cambiato schermo, se continueremo a scrivere in quello vecchio.

Un'altra cosa da fare: dobbiamo isolare la memoria necessaria per il secondo schermo. Ciò ridurrà ulte-

riormente i già scarsi 3500 byte a circa 3000, ma sarà un sacrificio necessario. Dopo tutto, non vogliamo che il Basic metta il naso in questa zona dello schermo, ma, a meno che non provvediamo in modo adeguato, il Basic userà tutta la memoria che trova libera.

Primo passo

Provate a battere PEEK(56) e premere RETURN. Dovreste vedere un 30 sullo schermo. Qualsiasi altro valore significa che il vostro VIC non ha i soliti 5 K di memoria, e il resto del discorso non ha valore.

Il valore 30 ci dice in quale "pagina" di memoria il Basic si è fermato. Una pagina è uno spezzone di 256 byte. Dobbiamo ora togliere altre due pagine per liberare abbastanza spazio per il secondo schermo.

Battete POKE 56,28: CLR seguito da RETURN. Ora abbiamo rubato circa 500 byte al Basic. Se non ci credete provate a scrivere PRINT FRE(0) e controllate. Non preoccupatevi troppo della perdita: ogni cosa tornerà al suo posto quando spegnerete la macchina; si tratta solo di un cambiamento momentaneo.

Per cambiare schermo

Ora provate a battere le seguenti quattro righe. Dovete scusare la congestione, ma l'inserimento degli spazi non avrebbe permesso di far stare tutto il testo e ci saremmo trovati a mezza via nel cambiamento di schermo. Scrivetele come un blocco unico senza premere RETURN:

```
POKE36866,22:POKE648,2
8:FORJ=217TO228:POKEJ,
156:NEXT:FORJ=229TO240
:POKEJ,157:NEXT
```

Rileggete attentamente; un errore e dovrete spegnere l'apparecchio e ricominciare da capo. Quando siete sicuri che è tutto a posto premete RETURN.

Oplà! Siete nell'altro schermo. Ha un aspetto un po' caotico, perché non è mai stato ripulito: eseguite un CLEAR e scrivete il vostro nome oppure qualcos'altro; potete anche cambiare colore ai caratteri. Ora per provare che abbiamo realmente due schermi, ritorniamo indietro.

Scrivete le prossime righe come un blocco unico:

```
POKE36866,150:POKE648,
30:FORJ=217TO228:POKEJ
,158:NEXT:FORJ=229TO25
0:POKEJ,159:NEXT
```

Controllate attentamente, premete RETURN, e siete di nuovo nel primo schermo.

Miglioramenti

La prima cosa che può darvi fastidio è che, appena cambiato schermo, il VIC scrive READY, probabilmente proprio in mezzo allo schermo nuovo. Questo non è un grosso problema; quando le istruzioni precedenti sono inserite in un programma, esse vengono eseguite senza che venga stampato alcun READY. Un altro vantaggio dell'inserire le istruzioni in un programma è che non dovete più condensare tutto in una sola riga.

Il secondo problema è meno appariscente, ma più serio. Le linee che precedentemente erano collegate, vengono spezzate; invece della nostra istruzione di quattro righe concatenate, abbiamo ora quattro righe ben separate. Ciò può non essere un problema per qualche tipo di messaggio e può addirittura essere piacevole se ottenuto intenzionalmente. Ma di solito vorremmo ritrovare lo schermo così come lo abbiamo lasciato.

Il trucco sta nelle locazioni da 217 a 240: per recuperare lo schermo, questi valori devono essere esattamente gli stessi di prima del nostro intervento. Questo richiede un po' di istruzioni aggiuntive.

Il programma

Ecco un breve programma per effettuare ciò di cui abbiamo bisogno. Il tasto F1 del VIC verrà usato per scambiare i due schermi.

```
100 REM  DOPPIO SCHERMO
110 POKE 56,28: CLR
120 DIM L%(23)
130 GOSUB 400: PRINT CHR$(147):
    GOSUB 400
140 Z$=CHR$(133)
200 GET X$: IF X$=Z$ THEN GOSUB 400
210 PRINT X$;: GOTO 200
400 REM  SCAMBIA
```

```
410 S=PEEK(648)
420 IF S=30 THEN S=28: T=22:
    GOTO 500
430 IF S=28 THEN S=30: T=150:
    GOTO 500
440 STOP: REM  ERRORE
500 POKE 648,S: POKE 36866,T
510 FOR J=0 TO 23
520 V=PEEK(J+217): POKE J+217,L%(J)
530 L%(J)=V
540 NEXT J
550 PRINT: RETURN
```

Ulteriori considerazioni

Così funziona. Si possono avere tre, quattro o cinque schermi? Sembra che non ci siano problemi. Sfortunatamente, bisogna scendere a un compromesso per realizzare una cosa simile, perché ci sono solo due tabelle dei colori dello schermo. Non ne abbiamo ancora parlato, perché nel nostro esempio esse si selezionano da sole scambiando i due schermi.

Omettendo l'istruzione PRINT alla riga 550, avremmo trovato un'altra stranezza: il VIC non cambia effettivamente schermo finché non completa una riga. Non cerca una nuova locazione dove mettere i caratteri finché un RETURN o qualcos'altro non segnala che una riga è finita. Abbiamo scelto la maniera più semplice in questo caso; per cambiare schermo in mezzo a una riga avremmo avuto bisogno di un programma molto più complesso.

Lasciamo questo esercizio allo stadio presente. È utile così com'è; non usa troppa memoria ed è abbastanza semplice.

Per quelli che vogliono capire il meccanismo: 36866 è la locazione che scambia effettivamente i due schermi. 648 dice al VIC dove si trova lo schermo. I valori da 217 a 240 hanno due compiti. Primo: dicono al VIC dove le righe dello schermo devono essere concatenate per formare un'unica riga. Ho chiamato questa serie di valori "tabella di concatenazione dello schermo", ma il nome è meno importante di quanto non sia capire il loro scopo. Le locazioni che controllano il colore sono nella parte alta della memoria, sopra 30000, e, per fortuna, non dobbiamo prenderle in considerazione in questo caso.

commodore

Effetti simultanei nei giochi di movimento

Ettore Massimo Albani

Su un PET/CBM serie 2000, 3000, 4000 e 8000 si provi a premere il tasto A e, tenendolo premuto, a battere il tasto Q. Si noterà che il carattere Q non viene visualizzato. Si provi ora a fare l'opposto, cioè a premere il tasto Q e, tenendolo premuto, a battere il tasto A. L'effetto sarà quello di ottenere la sequenza AQ ogni volta che si preme e si rilascia la A.

Come mai si ottengono effetti diversi nei due casi? La risposta risiede nel sistema operativo del PET.

Lo scopo di questa nota è comprendere il funzionamento della routine di scansione della tastiera e simulare in ambiente Basic una sua gestione. Naturalmente, l'interesse non è solo didattico, ma prevede la risoluzione di un grosso problema che si incontra nella stesura dei giochi utilizzanti la tastiera per muovere racchette o altro: l'impossibilità di premere più tasti per ottenere effetti simultanei sul video.

La tastiera

La tastiera degli elaboratori Commodore serie 2000, 3000, 4000 e 8000 può essere raffigurata, per quel che riguarda i suoi collegamenti elettrici, come una matrice a 10 righe ed 8 colonne, dove, in corrispondenza degli incroci, sono presenti dei tasti. Per esempio, premendo il tasto della lettera A, si chiude il circuito relativo alla quinta riga (riga 4) ed alla prima colonna (colonna 0), come indicato in fig. 1.

Si noti che degli 80 possibili tasti ne sono presenti solo 73 (o 76 nella tastiera della serie 8000 con i tasti TAB, ESC e REPEAT), non contando l'interruttore di SHIFT-LOCK inserito in parallelo allo SHIFT sinistro.

La gestione di interrupt

Ogni sessantesimo di secondo il microprocessore 6502 interrompe la sua normale attività per effettuare determinate operazioni, tra le quali vi è una particola-

	0	1	2	3	4	5	6	7
0	!		%	&	(	←	CLR HOME	⇔
1	"	$	/		)		⇕	INST DEL
2	Q	E	T	U	O	↑	7	9
3	W	R	Y	I	P		8	/
4	A	D	G	J	L		4	6
5	S	F	H	K	:		5	.
6	Z	C	B	M	;	RETURN	1	3
7	X	V	N	,	?		2	+
8	SHIFT SX.	@	]		>		0	−
9	OFF RVS	[	(SPACE)	<	RUN STOP	SHIFT DX.	.	=

Figura 1.

re procedura per sapere se è stato premuto un carattere sulla tastiera. Tale tipo di operazione viene detta *interrupt* ed è comandata dal clock del computer che 60 volte al secondo attiva una particolare linea del 6502 detta IRQ (*interrupt request*). Una volta attivata tale linea, il microprocessore memorizza nel suo registro stack l'attuale contenuto degli altri suoi registri: ciò gli consentirà di riprendere più tardi l'elaborazione interrotta. Successivamente, esamina le locazioni di memoria FFFE ed FFFF (65534 e 65535 in decimale) per prelevare l'indirizzo di partenza della subroutine di interrupt.

Come già detto, questa sobroutine esegue molte operazioni quali, ad esempio, l'aggiornamento dell'orologio TI, il rinfresco dei caratteri sul video, la scansione della tastiera, il controllo del bus IEEE-488, ecc.

Il 6520

Per la scansione della tastiera, il 6502 si avvale di un circuito integrato, il 6520, detto PIA (*Peripheral Interface Adapter*). Nel PET ne sono presenti due: il PIA 1 ed il PIA 2. Si tratta di un'interfaccia programmabile a due porte bidirezionali da 8 bit ciascuna, che possiede al suo interno 3 registri ad 8 bit per ogni porta; tuttavia, di questi 6 registri, solo 4 sono indirizzabili direttamente dal 6502. Ciò significa che il 6502 li "vede" come locazioni di memoria RAM (E810, E811, E812 ed E813; oppure 59408, 59409, 59410 e 59411 in decimale per il PIA 1). I primi due registri appartengono alla porta A, mentre gli altri due alla porta B. Inoltre, essi si dividono ulteriormente in registro di controllo e registro dati.

Il sistema operativo

Tornando al nostro problema, il sistema operativo del PET esegue la scansione della tastiera partendo dalla riga 9 e proseguendo verso la riga 0; ad ogni riga vengono esaminate le colonne, a partire dalla colonna 7 verso la colonna 0. Se durante la lettura della riga X il sistema operativo rileva che è stato premuto il tasto relativo alla colonna Y, esso fa ripartire la scansione dalla colonna 7, ripetendo questa operazione per 10 volte prima di accettare il carattere corrispondente (questo per eliminare disturbi di vario genere). Inoltre, il sistema operativo non accetta per più di una volta lo stesso carattere (questo ad evitare che, tenendo premuto il tasto relativo, esso venga visualizzato più volte). Si intuisce che, tuttavia, se viene premuto un tasto con un numero di riga o di colonna maggiore di X o Y rispettivamente, il sistema operativo accetta il nuovo carattere e, dopo le 10 scansioni, lo visualizza. Immediatamente dopo averlo rilasciato, però, viene riconsiderato il tasto (X,Y) ancora premuto e, dopo i soliti 10 tentativi, viene visualizzato.

Se, invece, viene premuto un tasto con un numero di riga o di colonna inferiore ad X o Y rispettivamente, il sistema operativo non si accorge della cosa, poiché la scansione non può arrivare ad esso, essendo bloccata tra la riga 9 e la riga X e tra la colonna 7 e la colonna Y. Questo è, appunto, il caso del carattere Q che non può essere visualizzato finché rimane premuta una A.

La simulazione

Come fa il PIA 1 a gestire la scansione della tastiera? Il funzionamento risulta abbastanza semplice: i 4 bit meno significativi della porta A, configurata come

porta di uscita, sono connessi ad un circuito integrato decodificatore, le cui uscite sono a loro volta connesse con le righe della matrice-tastiera (vedi fig. 2). La porta B è, invece, configurata come porta di ingresso e i suoi 8 bit sono direttamente connessi con le colonne. Per ottenere la scansione, il sistema operativo non fa altro che decrementare da 9 a 0 i 4 bit della porta di uscita A, osservando ciò che succede ai bit della porta di ingresso B. La decodifica, che ha le uscite a livello normalmente alto (+5 V), porta a livello basso (circa +0.5 V) solo l'uscita relativa alla combinazione in codice BCD presente all'ingresso. Premendo un pulsante della riga in esame, si ottiene come risultato quello di portare a livello basso la colonna relativa. Normalmente, le colonne sono tutte a livello alto, perciò, premendo ad esempio il tasto A, non appena i 4 bit della porta A segnano 0100 (4 in decimale), il registro dati della porta B da 11111111 diventa 11111110.

Esiste la possibilità di disattivare la scansione automatica della tastiera agendo sul PIA 1, cioè si pone a zero il bit meno significativo del registro di controllo della seconda porta; ciò significa che occorre portare il contenuto della locazione E813 (59411), che normalmente è 00111101, a 00111100.

Il programma di fig. 3 è una simulazione in Basic del processo di scansione. L'unico neo di tale routine è dato dalla velocità limitata. Si noti anche che il numero di tentativi di lettura-colonna è stato limitato a due e che la linea 110, oltre a "spegnere" la scansione, inibisce anche l'interpretazione del tasto di STOP: ecco perché è stata inserita la linea 310. Il funzionamento del programma è elementare; facendolo partire, ogni volta che viene premuto un tasto appaiono sul video i valori delle relative riga e colonna.

Un'applicazione più interessante è quella che sfrutta questa tecnica per sostituire l'istruzione Basic di GET X$. Come esempio, si è scelto il programma "Ostacoli", prelevato dal libro *32 programmi con il PET*, edito da Franco Muzzio editore, Padova. Un listato del programma è riportato in fig. 4. Il gioco, per due persone, consiste nel chiudere l'avversario in un ipotetico muro, la cui costruzione viene diretta manovrando opportunamente ben 8 tasti. Infatti, ogni giocatore dispone di 4 direzioni per costruire il muro, secondo lo schema di fig. 5.

Come si osserva dal listato, la routine per prelevare il carattere inizia alla linea 255 e termina alla linea 340. È evidente che la routine accetta un solo carattere, per cui, se uno dei due giocatori tiene premuto un tasto, può inibire il funzionamento di quelli del suo

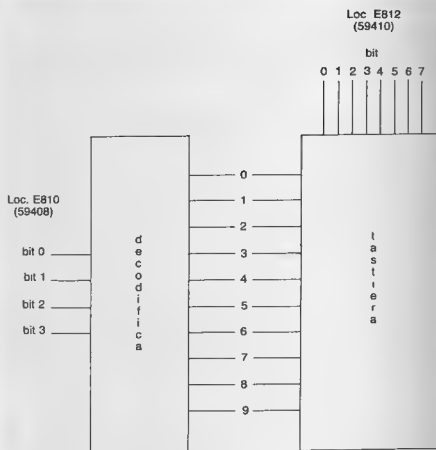



Figura 2.

```
100 REM * KEYSCAN
105 REM * DISABILITA INTERRUPT
110 POKE 59411,60: R=-9: C=-9: YY=-9
115 REM * SCANSIONE RIGHE
120 FOR X=9 TO 0 STEP -1
125 REM * ABILITA LA X-ESIMA RIGA
130 POKE 59408,240+X
135 REM * SCANSIONE COLONNA
140 K=PEEK(59410): IF K=255 THEN 290
150 IF (K AND 128)=0 THEN Y=7: GOTO 230
160 IF (K AND 64)=0 THEN Y=6: GOTO 230
170 IF (K AND 32)=0 THEN Y=5: GOTO 230
180 IF (K AND 16)=0 THEN Y=4: GOTO 230
190 IF (K AND 8)=0 THEN Y=3: GOTO 230
200 IF (K AND 4)=0 THEN Y=2: GOTO 230
210 IF (K AND 2)=0 THEN Y=1: GOTO 230
220 IF (K AND 1)=0 THEN Y=0
225 REM * RIPETIZIONE ANTI DISTURBO
230 IF YY<>Y THEN N=0: YY=Y
240 N=N+1: IF N=2 THEN N=0: GOTO 260
250 GOTO 140
255 REM * CONTROLLO VALIDITA' CARATTERE
260 IF R=X AND C=Y THEN M=0: GOTO280
270 R=X :C=Y: PRINT "RIGA"R"COLONNA"C
280 X=0
290 NEXT X
295 REM * DOPO 4 CICLI DIMENTICA
300 M=M+1: IF M=4 THEN M=0: R=-9: C=-9
305 REM * E' STATO PREMUTO LO STOP ?
310 IF R=9 AND C=4 THEN 330
320 GOTO 120
330 PRINT "STOP": POKE 59411,61
```

Figura 3.

sinclair Tandy sirius

BMC VIC-20 CASIO

HANIMEX SEIKOSHA GRUPPO EDITORIALE JACKSON

DAI THE MICROCOMPUTER COMPANY TEXAS INSTRUMENTS ATARI

SHOP bit primavera

ALESSANDRIA Via Savonarola, 13

ANCONA Via De Gasperi, 40

AREZZO Via F. Lippi, 13

BARI Via Capruzzi, 192

BARI Via Devitofrancesco, 4/2 A

BASSANO DEL GRAPPA Via Jacopo Da Ponte, 51

BERGAMO Via F. D'Assisi, 5

BOLOGNA Via Brugnoli, 1

CAGLIARI Via Zagabria, 47

CAMPOBASSO Via Mons. II Bologna, 10

CESANO MADERNO Via Ferrini, 6

CINISELLO BALSAMO Viale Matteotti, 66

COMO Via L. Sacco, 3

COSENZA Via Dei Mille, 86

FAVRIA CANAVESE Corso Matteotti, 13

FIRENZE Via G. Milanesi, 28/30

FOGGIA Via Marchianò, 1

FORLI Piazza M. Degli Ambrogi, 1

GALLARATE Via A. Da Brescia, 2

GENOVA Via D. Fiasella, 51/R

GENOVA-SESTRI Via Chiaravagna, 10/R

IMPERIA Via Delbecchi, 32

L'AQUILA Via Strada 85 N°2

LECCO Via L. Da Vinci, 7

LIVORNO Via San Simone, 31

MESSINA Via Del Vespro, 71

MILANO Galleria Manzoni, 40

MILANO Via Petrella, 6

MILANO Via Cantoni, 7

MILANO Piazza Firenze, 4

MILANO Via Altaguardia, 2

MILANO Viale Corsica, 14

MONZA Via Azzone Visconti, 39

NAPOLI Via Luigia Sanfelice, 7/A

NOVARA Baluardo Q. Sella, 32

PADOVA Via Fistomba, 8

PALERMO Via Lamarmora, 82

PAVIA Via C. Battisti, 4/A

PARMA Via Imbriani, 41

PARMA Via Borghesi, 16

PERUGIA Via R. D'Andreotto, 49/55

PESCARA Via Guelfi, 74

PIACENZA Via IV Novembre, 60

PISA Via XXIV Maggio, 101

PISTOIA Viale Adua, 350

POTENZA Via Mazzini, 72

POZZUOLI Via Pergolesi, 13

RIMINI Via Bertola, 75

ROMA Via C. Da Spoleto, 23

ROMA Piazza S. Dona Di Piave, 14

ROMA Viale Quattro Venti, 152

ROMA Largo Belloni, 4

TERAMO Via Martiri Pennesi, 14

TERNI Via Beccaria, 20

TORINO Via Chivasso, 11

TORINO Corso Grosseto, 209

TORINO Via Tripoli, 179

TRENTO Via N. D Arco, 15/2

TREVIGLIO Via Mazzini, 10/B

TRIESTE Via F. Saverio, 138

VERONA Via Pontiere, 2

VARESE Via Carrobio, 13

VIAREGGIO Via A. Volta, 79

VOGHERA Piazza Carducci, 11

SONDRIO Via N. Sauro, 28

La prima e la piú grande catena di computer in Italia.

avversario, con le priorità stabilite dalla matrice di fig. 1.

In fig. 6 vi è la routine modificata. La linea 253 serve a rallentare la costruzione del muro, mentre le linee 157 e 158 sono state aggiunte per l'assegnazione delle variabili.

W

A ← → D

X

giocatore di sinistra

8

4 ← → 6

2

giocatore di destra

Figura 5.

```
100 REM:OSTACOLI
110 REM:COPYRIGHT 1979 BY TOM RUGG AND PHIL FELDMAN
120 X=RND(TI):GOSUB 600
125 PRINT CHR$(147):PRINT
130 PRINT TAB(15);CHR$(18);"OSTACOLI"
140 PRINT:PRINT
150 PRINT"  BATTI UN TASTO QUALUNQUE PER PARTIRE"
155 Z=90: R3=243: R4=244: R7=247
157 C0=1: C1=2: C2=4: C3=8: C4=16: C5=32: C6=64: C7=128
158 PA=59408: PB=59410: CB=59411: IN=60: NI=61: NL=255
160 AX=10:AY=12:BX=29:BY=12:A=86:B=102
165 S=32768:E=127:AD=INT(4*RND(1))+1
167 BD=INT(4*RND(1))+1
170 GOSUB 900:GOSUB 950
180 GET R$:IF R$="" THEN 180
190 PRINT CHR$(147);
200 GOSUB 950:GOSUB 900
205 FOR J=1 TO 10:GET R$:NEXT
210 X=AX:Y=AY:D=AD:GOSUB 1000
220 AR=R:AX=X:AY=Y
230 X=BX:Y=BY:D=BD:GOSUB 1000
240 BR=R:BX=X:BY=Y
245 IF AR=1 OR BR=1 THEN 400
250 GOSUB 900
253 FOR J=1 TO 50: NEXT
255 POKE CB,IN
260 POKE PA,R3: K=PEEK(PB): IF K=NL THEN 275
265 IF (K AND C0)=0 THEN AD=1
270 IF (K AND C6)=0 THEN BD=1
275 POKE PA,R4: K=PEEK(PB): IF K=NL THEN 300
280 IF (K AND C0)=0 THEN AD=3: GOTO 290
285 IF (K AND C1)=0 THEN AD=4
290 IF (K AND C6)=0 THEN BD=3: GOTO 300
295 IF (K AND C7)=0 THEN BD=4
300 POKE PA,R7: K=PEEK(PB): IF K=NL THEN 315
305 IF (K AND C0)=0 THEN AD=2
310 IF (K AND C6)=0 THEN BD=2
315 POKE CB,NI
350 GOTO 210
400 GOSUB 700:X=AX:Y=AY
410 IF BR=1 THEN X=BX:Y=BY
420 FOR J=1 TO 15
430 POKE S+40*Y+X,Z
440 FOR K=1 TO 200:NEXT
450 POKE S+40*Y+X,Z+128
460 FOR K=1 TO 200:NEXT
470 NEXT
480 FOR J=1 TO 20:GET R$:NEXT
490 GOTO 125
600 PRINT CHR$(147)
610 INPUT"NOME DEL GIOCATORE DI SINISTRA";A$
620 PRINT
630 INPUT"GIOCATORE DI DESTRA";B$
640 RETURN
700 PRINT CHR$(19);
710 FOR J=1 TO 12
720 PRINT CHR$(17):NEXT
730 IF AR=1 AND BR=1 THEN PRINT"AVETE PERSO ENTRAMBI  ";:RETURN
740 R$=A$:IF AR=1 THEN R$=B$
750 PRINT "VINCE ";R$;"  ";
760 RETURN
900 POKE (S+40*AY+AX),A
910 POKE (S+40*BY+BX),B
920 RETURN
950 FOR X=0 TO 39
960 POKE S+X,E:POKE S+880+X,E
970 NEXT:FOR Y=0 TO 22
980 POKE (S+40*Y),E:POKE (S+40*Y+39),E:NEXT
990 RETURN
1000 IF D=1 THEN Y=Y-1
1010 IF D=2 THEN Y=Y+1
1020 IF D=3 THEN X=X-1
1030 IF D=4 THEN X=X+1
1040 R=0
1050 IF PEEK(S+40*Y+X)<>32 THEN R=1
1060 RETURN

READY.
```

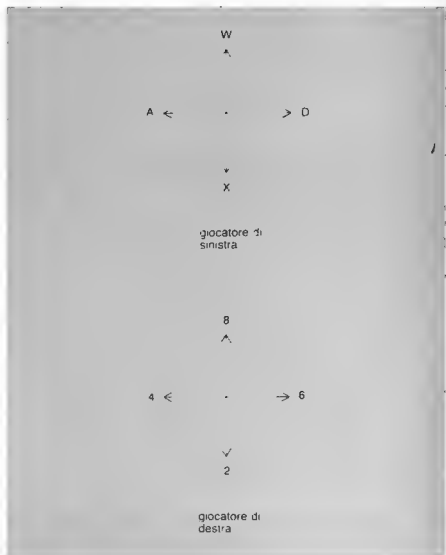



Figura 4.

```
157 R3=243: R4=244: R7=247: C1=2: C2=4: C3=8: C4=16: C5=32  C6=64: C7=128
158 PA=59408: PB=59410: CB=59411: IN=60: NI=61: NL=255
```

```
250 GOSUB 900
253 FOR J=1 TO 50: NEXT
255 POKE CB,IN
260 POKE PA,R3: K=PEEK(PB): IF K=NL THEN 275
265 IF (K AND C0)=0 THEN AD=1
270 IF (K AND C6)=0 THEN BD=1
275 POKE PA,R4: K=PEEK(PB): IF K=NL THEN 300
280 IF (K AND C0)=0 THEN AD=3: GOTO 290
285 IF (K AND C1)=0 THEN AD=4
290 IF (K AND C6)=0 THEN BD=3: GOTO 300
295 IF (K AND C7)=0 THEN BD=4
300 POKE PA,R7: K=PEEK(PB): IF K=NL THEN 315
305 IF (K AND C0)=0 THEN AD=2
310 IF (K AND C6)=0 THEN BD=2
315 POKE CB,NI
350 GOTO 210
```

Figura 6.

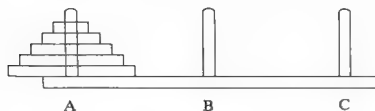
è in edicola il nuovo numero

- IL BOOM DEI SISTEMI FAULT-TOLERANT
- UNIX FROM BERKELEY
- PABX: CARATTERISTICHE ED EVOLUZIONE FUNZIONALE
- SIP-TELEMATICA: IL PUNTO SULLA SITUAZIONE
- IN ARRIVO IL SUPERMINI HP: UN 32 BIT CHE FARA' PARLARE DI SÈ

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

La torre di Hanoi

Mauro Boscarol

La torre di Hanoi è un vecchio gioco di tipo logico-matematico. Racconta Martin Gardner nel volume 1 di *Enigmi e giochi matematici* (Sansoni), che fu inventato da un matematico francese, Edouard Lucas, che lo mise in vendita nel 1883.

In figura è rappresentata la situazione iniziale del gioco. Il problema è trasferire gli otto dischi della torre dal piolo in cui sono attualmente (A) al piolo C, rispettando le seguenti regole:

(a) si può spostare solo un disco alla volta;
(b) non si può mettere un disco più grande sopra ad uno più piccolo.

La descrizione originale del gioco parla della mitica "torre di Brahma" esistente un tempo nella città di Benares. Questa torre, diceva la descrizione, consiste di 64 dischi d'oro, che attualmente i sacerdoti del tempio stanno spostando. Prima che essi possano portare a termine il loro compito, si diceva, il mondo scomparirà.

A parte questa suggestiva descrizione, la soluzione del gioco si può trovare con un algoritmo di tipo ricorsivo. Ne risulta un bell'esempio di applicazione di questa tecnica e un classico esercizio scolastico sulla ricorsività.

Per risolvere il problema per n cerchi, conviene immaginare di saperlo fare per $n-1$ cerchi. In tal caso si possono spostare i primi $n-1$ cerchi da A a B, poi il cerchio rimasto da A a C e infine gli $n-1$ cerchi da B a C. Indicando con n: A→B lo spostamento di n cerchi da A a B, il problema si risolve come in figura 1, dove n: A→C significa spostare n cerchi da A a C.

Resta il fatto che abbiamo immaginato di saperlo fare per $n-1$, ma in effetti non lo sappiamo fare. Immaginiamo allora di saperlo fare per $n-2$ dischi.

$$n : A \rightarrow C \begin{cases} n-1 : A \rightarrow B \\ \quad 1 : A \rightarrow C \\ n-1 : B \rightarrow C \end{cases}$$

Fig. 1.

In tal caso, per spostare $n-1$ dischi da A a B si fa come in figura 2, e per spostare $n-1$ dischi

$$n-1 : A \rightarrow B \begin{cases} n-2 : A \rightarrow C \\ \quad 1 : A \rightarrow B \\ n-2 : C \rightarrow B \end{cases}$$

Fig. 2.

$$n-1 : B \rightarrow C \begin{cases} n-2 : B \rightarrow A \\ \quad 1 : B \rightarrow C \\ n-2 : A \rightarrow C \end{cases}$$

Fig. 3.

da B a C si può fare come in figura 3 e quindi la figura 1 si può completare nella figura 4.

$$n : A \rightarrow C \begin{cases} n-1 : A \rightarrow B & \begin{cases} n-2 : A \rightarrow C \\ \quad 1 : A \rightarrow B \\ n-2 : C \rightarrow B \end{cases} \\ \quad 1 : A \rightarrow C & \\ n-1 : B \rightarrow C & \begin{cases} n-2 : B \rightarrow A \\ \quad 1 : B \rightarrow C \\ n-2 : A \rightarrow C \end{cases} \end{cases}$$

Fig. 4.

Naturalmente per $n-2$ dischi non sappiamo farlo, ma possiamo immaginare di saperlo fare per $n-3$ dischi, e così via fino a che si arriva ad un disco: questo sappiamo farlo veramente. Per esempio, se $n=4$, lo schema completo è quello della figura 5, e quindi le mosse da fare sono quelle che si leggono dall'alto verso il basso: A→B, A→C, B→C, A→B, C→A, C→B, A→B, A→C, B→C, B→A, C→A, B→C, A→B, A→C, B→C.

Si può anche tenere a mente una regola: una

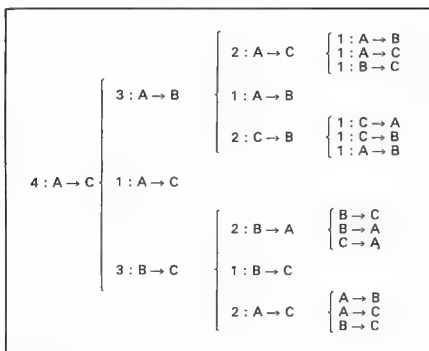



Fig. 5.

volta sì e una volta no si muove il disco più piccolo in modo circolare. Quando non si fa questa mossa, si fa l'unica altra possibile.

Si può controllare facilmente che le mosse per spostare n dischi sono 2^n-1. I sacerdoti del tempio, per spostare 64 dischi, ci avrebbero messo $2^{64}-1$ mosse. Si tratta di un numero di 20 cifre:

18 446 744 073 709 551 615

Se anche riuscissero a spostarne uno al minuto sarebbero sempre trenta miliardi di millenni, ed effettivamente in così tanto tempo il mondo ha molte probabilità di scomparire.

Noi però possiamo fare un po' meglio, facendo lavorare un calcolatore.

Il listato Basic qui riportato risolve il gioco per n dischi senza utilizzare procedure ricorsive (che del resto nel Basic non sono implementate) né simulandole.

Eseguendo questo programma, ricordatevi dei sacerdoti del tempio. Un computer non ci mette un minuto a spostare un disco, questo è vero, però non ci mette poi tanto meno: diciamo 100 volte di meno. Per n=64 sono trecento milioni di millenni.

```
10 INPUT"NUMERO DI DISCHI";N
20 I=1
30 J=3
40 GOSUB100
50 END
100 IF N=0 THEN RETURN
110 N=N-1
120 J=6-I-J
130 GOSUB100
140 J=6-I-J
150 PRINT"SPOSTA UN CERCHIO DA"I"A"J
160 I=6-I-J
170 GOSUB 100
180 I=6-I-J
190 N=N+1
200 RETURN
```

Regole di stile per autori di software

Nino Tonolli

Sebbene gli autori abbiano riferito queste note alle applicazioni di tipo didattico, i loro suggerimenti possono essere applicati a tutti i tipi di programmi.

Sulla scia del rapido sviluppo della tecnologia dei computer, i creatori di software devono soddisfare la richiesta di programmi per ogni nuovo computer. Esiste in questo momento una abbondanza di software. Solo una piccola percentuale è di tipo educativo e la qualità dei programmi varia in modo notevole.

Questo dipende in larga parte dal fatto che i programmi di tipo educativo sono stati scritti da hobbisti, educatori, editori, aziende di software o aziende che producono computer. Non esiste un formato e una qualità standard, né coerente. Ciò rende i programmi difficili da usare e da capire.

Quella che segue è una lista di standard di programma, stilata nella Lawrence Hall of Science. Essa può servire da guida per chi deve scrivere o modificare programmi per uso sia privato che dimostrativo.

Per la maggior parte questi standard sono indipendenti dal particolare computer utilizzato e possono essere applicati a programmi scritti per qualsiasi macchina.

L'inizio del programma

1. Scrivete delle righe di commento (REM) all'inizio del programma per identificarlo e descriverlo. La documentazione esterna può essere persa, così è necessario che il programma stesso la contenga al suo interno. Le istruzioni REM devono fornire le seguenti informazioni:

(a) nome del programma,
(b) descrizione del programma in una riga,
(c) tipo di computer per cui è scritto (se necessario, il modello o la versione di ROM),
(d) occupazione minima di memoria (in Kbyte) richiesta per l'esecuzione, compresa l'area occupata dalle variabili,
(e) linguaggio o versione di linguaggio (es. Integer Basic o Applesoft Basic),
(f) eventuali periferiche necessarie (es. joystick, palette, stampante, disco ecc.),
(g) quanti e quali file vengono usati (es. sull'Apple, quali file sorgente o in linguaggio macchina o subroutine sono letti da file),
(h) autore o azienda produttrice,
(i) indirizzo dell'autore o azienda,
(j) data di stesura del programma,
(k) indicazioni di copyright o permesso di duplicazione,
(l) modifiche apportate alla versione base,
(m) autore delle modifiche,
(n) data dell'ultima modifica apportata.

2. Inizializzate il computer secondo lo scopo del programma. Per esempio se dovete usare la grafica e le maiuscole del PET, il computer deve essere predisposto per ciò. Sull'Apple dovrete invece predisporre la pagina di testo se avete bisogno di scrivere sul video. Non date per scontato che il computer sia già predisposto nel modo corretto.

3. Cancellate lo schermo, così i resti di altri programmi non saranno visibili dagli utilizzatori del programma.

4. Fate scrivere il nome del programma insieme ai convenevoli (es. "Benvenuti al gioco della cava"). Questo permette all'utilizzatore di sapere che tipo di programma sta girando in quel momento.

5. Chiedete all'utilizzatore se desidera vedere le istruzioni. Se è già esperto del programma può voler saltare le istruzioni.

Istruzioni

6. Spaziate le righe di spiegazione, così saranno più facili da leggere.

7. Contenete le istruzioni in tre pagine di testo.

8. Usate diagrammi di spiegazione ovunque possibile.

9. Scegliete un carattere (per es. I per indietro) che permetta all'utente di tornare indietro alla parte di istruzioni che precede quella visualizzata.

10. Dopo aver fornito le istruzioni, chiedete all'utilizzatore se intende effettivamente continuare con il programma (es. "Vuoi giocare?" "Se sei pronto premi un tasto"). Dopo aver letto le istruzioni, può aver deciso di non provare il programma. Se non volete permettere questa scelta dell'utente, saltate questo passo.

11. Includete nel programma dei richiami al modo di giocare. Per esempio, se l'utente deve introdurre delle coordinate, il programma può ricordargli di scrivere due numeri separati da una virgola.

Input

12. Usate routine di input piuttosto che istruzioni INPUT. L'istruzione Basic INPUT ha parecchi difetti che possono essere evitati. La routine dovrebbe:
(a) pulire il buffer della tastiera prima di accettare i dati; caratteri eventualmente già presenti possono essere dovuti a errori e devono perciò essere ignorati,
(b) evitare che l'utente possa fermare il programma premendo il tasto RETURN senza aver inserito i dati,
(c) evitare che l'utente possa cancellare troppi caratteri correggendo i dati scritti (per es. l'utente non deve poter cancellare il testo della domanda),
(d) eliminare l'effetto dei caratteri che controllano il cursore o che cancellano lo schermo, battuti dall'utente;
(e) limitare il numero di caratteri che possono essere immessi; per esempio, se volete che siano inserite non più di nove cifre, l'utilizzatore non deve poter scrivere più dei nove caratteri necessari,
(f) permettere che il cursore possa essere spostato a sinistra di una posizione, per cancellare un carattere alla volta quando l'utente corregge i suoi errori; i caratteri a destra del cursore non devono essere trascinati a sinistra nelle operazioni di correzione.

13. Fate aspettare il programma finché l'utente non preme il tasto RETURN dopo aver introdotto i dati, piuttosto che continuare col programma appena viene premuto un tasto.

14. Verificate che i dati introdotti non siano errati. Se viene commesso un errore, spiegatene il tipo e chiedete di nuovo i dati all'utente. Per esempio, se un utente batte una lettera al posto di un numero, un messaggio di errore potrebbe essere: "Ricordati, puoi usare solo numeri"

15. Lasciate abbastanza spazio dopo ogni domanda (per esempio: "Come ti chiami?") per poter battere la risposta senza andare nella prossima riga.

16. Nelle domande SI/NO, controllate solo la prima lettera della risposta. Se questa non è una S o una N, chiedete all'utente di scrivere SI o NO e ripetete la domanda. Questo permette all'utilizzatore di rispondere con una sola lettera o con l'intera parola.

17. Controllate che non si verifichino errori nel programma a causa dei dati forniti. (Per esempio una divisione per zero, un superamento dello spazio disponibile per le stringhe, ecc.) Eventualmente date un messaggio di errore per l'utente al momento dell'input.

Nel programma

18. Organizzate il testo in pagine anziché farlo scorrere verso l'alto, oppure usate delle temporizzazioni per permettere all'utente di leggere comodamente. Si deve:
(a) per prima cosa cancellare lo schermo,
(b) stampare una pagina di testo facendo attenzione che non scorra verso l'alto,
(c) riservare l'ultima riga di testo di ogni pagina per poter stampare eventuali messaggi di errore o del tipo "Premi lo spazio per continuare".

Alla fine del programma

25. Al termine del programma chiedete all'utente se vuole riprovare. Permettetegli di modificare le condizioni iniziali (per esempio il livello di difficoltà o il limite massimo di tempo) se lo desidera.

26. Se l'utente ha terminato il programma o lo interrompe, questo dovrebbe:
(a) pulire lo schermo,
(b) dare un giudizio sulle prestazioni dell'utente, se è il caso,
(c) finire, lasciando i commenti finali sullo schermo e posizionando il cursore all'inizio di una riga sgombra per permettere all'utilizzatore di inserire un comando, oppure:
(d) chiedere all'utente di premere lo spazio e dopo aver ricevuto conferma, far partire un "menù" che faccia la lista dei programmi disponibili.

Gli standard che abbiamo descritto sono un tentativo di dare dei suggerimenti per scrivere programmi che si spieghino da soli, facili da capire e da usare, e relativamente privi di errori. Accettiamo volentieri eventuali suggerimenti o aggiunte.

Imparare il Basic dei personal

Giuseppe Staluppi

Quasi due anni fa, nel gennaio dell'81, l'editore Franco Muzzio di Padova iniziava a pubblicare una serie di libri specificamente rivolti agli utenti dei personal computer. In questo periodo di tempo la collana è cresciuta ed ora conta una quindicina di volumi. Si tratta di testi dedicati ai personal computer in generale, al Basic, al PET/CBM, all'Apple, al TRS-80, al Pascal, al CP/M e ad altri argomenti tutti connessi con i personal computer e le loro applicazioni.

Credo che i titoli più rappresentativi della collana siano i due volumi intitolati *Il Basic e il personal computer*, il primo dedicato ad una introduzione generale, il secondo dedicato alle applicazioni.

Caratteristica di questi volumi è la loro generalità: possono essere letti da chiunque e vanno bene per qualunque tipo di personal computer.

Il primo volume comprende un breve ma completo corso di Basic: in otto lezioni si impara a conoscere l'argomento e si inizia a programmare in questo linguaggio. I primi, semplici programmi che vengono poi proposti sono elementari esercizi di grafica: grafici di funzioni, istogrammi.

Il secondo volume entra nel vivo delle applicazioni. Numerose pagine sono dedicate ai metodi di ordinamento, dai più semplici ai più complessi. Un'altra sezione è dedicata ai giochi: giochi su scacchiera, giochi d'azzardo, giochi dinamici, giochi di simulazione. Si parla poi di *computer art*, di basi di dati, di simulazione.

Thomas Dwyer e Margot Critchfield
Il Basic e il personal computer
uno: introduzione
198 pagine, 9.500 lire
Il Basic e il personal computer
due: applicazioni
214 pagine, 14.000 lire
Padova: Franco Muzzio & c. editore, 1982

Complessivamente, il materiale presentato è molto abbondante e piacevole. Ma la cosa che maggiormente colpisce il lettore è lo stile amichevole e informale con cui i due libri sono scritti. Le numerose illustrazioni e i numerosissimi listati di programmi, pronti per essere inseriti nel vostro personal, rendono la lettura piacevole e l'apprendimento sicuro.

Se siete alla ricerca di un modo facile ed attraente di imparare il Basic, questi due volumi fanno al caso vostro.

Ricerca su alberi

Seconda parte: Tecniche euristiche

di *Gregg Williams*

I metodi esaustivi di ricerca su alberi, per motivi di cui parleremo in seguito, trovano *alla fin fine* il cammino ottimale dal nodo di partenza *S* alla meta più vicina ad esso. L'espansione di tipo esponenziale di molti problemi può superare le disponibilità di memoria e di tempo anche dei computer più grandi; per questo sono stati studiati metodi che limitano il numero di nodi estesi, considerando però i nodi che conducono alla meta più vicina. Queste *tecniche euristiche* raccolgono informazioni da un nodo e le usano per determinare la probabilità di trovarsi sul cammino minimo verso una meta.

In questo articolo avremo modo di trattare due tipi di tecniche euristiche, *accettabili* e *non accettabili*, e le proveremo utilizzando il programma Basic della prima parte dell'articolo.

Teoria per l'algoritmo accettabile

Una strategia di ricerca sull'albero di un problema consiste nell'ordinare la lista dei nodi non estesi assegnando a ciascun nodo un valore numerico e conferendo al programma la capacità di scegliere come nodo da estendere quello col valore più basso (è il metodo usato dal programma SEARCH nella prima parte di questo articolo). Sebbene l'algoritmo possa agire con qualsiasi ordinamento che porti al risultato corretto, una piccola restrizione al genere dell'ordinamento dà luogo ad un algoritmo di ricerca che trova sicuramente sia una meta che la meta ottimale — cioè la meta con *costo* minore. Un algoritmo di questo tipo si dice *accettabile*.

Consideriamo l'albero parziale della figura 1. (Supporremo che i cammini da *S* ad *n* e da *n* a *G* siano i percorsi minimi). Definiamo con $g(n)$ il cammino minimo dal nodo iniziale *S* ad *n*; e con $h(n)$ il cammino minimo da *n* alla meta più vicina *G*. Allora

$$f(n) = g(n) + h(n)$$

è il costo del percorso ottimale verso una meta, passando per il nodo *n*. (Se questo cammino non esiste, il costo si dice indefinito; in un programma, alla variabile di costo relativa verrebbe assegnato un valore alto arbitrario.)

Ora che disponiamo delle tre funzioni *f*, *g* ed *h*, ne definiamo altre tre, $\hat{f}$ (da leggere "*f*-cappello"), $\hat{g}$ ed $\hat{h}$, che, in una data situazione, sono stime delle funzioni minime teoriche (spesso sconosciute) *f*, *g* ed *h*. In altri termini, $\hat{f}(n)$ è il costo stimato per il percorso minimo da *S* a *G* attraverso *n*; $\hat{g}(n)$ è il costo stimato del cammino minimo da *S* ad *n* (ricordate che un percorso da *S* ad *n* può non essere *minimo*); ed $\hat{h}(n)$ è il costo stimato del cammino minimo da *n* alla meta più vicina (che è, al momento, ignota).

La riproduzione di questo articolo è stata concessa da BYTE. Traduzione di Flavio Santini

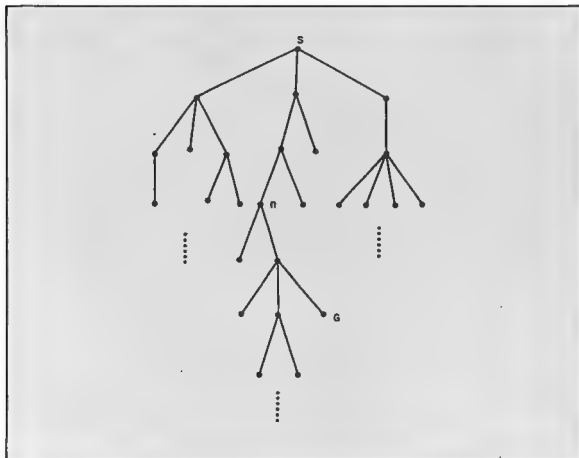



Fig. 1. *Un pezzo di albero. In questa rappresentazione, ogni stato (nodo) del problema è dato da un punto ed ogni ramo rappresenta il passaggio da uno stato a quello successivo. Il nodo* S *è il problema originale, mentre il nodo* n *e tutti gli altri sono tappe intermedie sulla via verso la soluzione (il nodo* G*, la meta).*

Intuitivamente, senza dimostrazione, la condizione necessaria perché un algoritmo che trova $\hat{h}(n)$ sia accettabile è che il metodo d'ordinamento dia luogo ad un valore sicuramente minore o uguale, per ogni nodo n, al costo del percorso minimo che va da n alla meta più vicina. Formalmente la condizione si esprime così:

$$\hat{f}(n) \leq f(n)$$

Se questa condizione è sempre verificata, allora l'algoritmo di ordinamento è accettabile. (I lettori interessati alla dimostrazione la troveranno in *Problem Solving Methods in Artificial Intelligence*, di Nils Nilsson, 1971, pagg. 59-65.)

Consideriamo due algoritmi che si riconoscano facilmente come accettabili. Il primo è quello del metodo *breadth-first*, che non dà informazioni sul valore relativo ad ogni nodo — cioè $\hat{h}(n) = 0$. (Nota: il programma sviluppato nella parte 1 utilizzava un valore diverso per la variabile di $\hat{h}(n)$, *D1*, a scopo dimostrativo; comunque, $D1 = 0$ darà lo stesso risultato.) Poiché zero è un limite inferiore per il costo minimo relativo a qualsiasi nodo, meta o non meta (cioè $0 \leq h(n)$), alla luce della disuguaglianza di cui sopra l'algoritmo breadth-first appare accettabile. Tuttavia, come sappiamo per esperienza, l'algoritmo breadth-first è non selettivo; cioè estende tutti i nodi in ordine crescente di livello finché trova la prima (e perciò minimale) meta. Così la totale assenza di informazioni euristiche accompagna e misura la sua inefficienza.

D'altro canto, consideriamo un algoritmo d'ordinamento $\hat{h}$ che dia *esattamente* il costo del percorso minimo da n a G; in altre parole, $\hat{h}(n) = h(n)$ per tutti gli n, il che soddisfa ancora la disuguaglianza precedente. Che significato ha? Riflettendo un attimo vi convincerete che, primo, poiché questo algoritmo fornisce informazioni complete sullo stato del sistema, raggiungerà sicuramente la meta più vicina G; secondo, lo farà senza estendere alcun nodo superfluo. Cosa c'è di più semplice? Siccome l'algoritmo di ricerca estende sempre il nodo con il più piccolo valore di $\hat{h}$, e siccome in questo caso il valore di $\hat{h}$ è *esattamente* il costo del percorso da quel nodo alla meta, l'algoritmo di ricerca, ad ogni estensione, si avvicinerà di un nodo alla meta. Perciò la presenza di informazioni euristiche complete corrisponde alla massima efficienza.

A partire dagli estremi

$$\hat{h}(n) = 0$$

e

$$\hat{h}(n) = h(n)$$

ci aspetteremmo di trovare un $\hat{h}(n)$ che soddisfa

$$0 < \hat{h}(n) < h(n)$$

intermedio tra i due limiti di efficienza, con efficienza proporzionale all'avvicinarsi di $\hat{h}(n)$ ad $h(n)$, per tutti i nodi n. Ed in effetti è così: dati due algoritmi d'ordinamento accettabili A (che genera $\hat{h}(n)$) ed A^* (che genera $\hat{h}^*(n)$), si dice che A^* *dà più informazioni* di A se $\hat{h}^*$ è sempre maggiore o uguale di $\hat{h}$, oppure:

$$\hat{h}(n) \leq \hat{h}^*(n) \leq h(n).$$

È stato anche dimostrato che A^* estende con certezza un numero di nodi minore o uguale di quello di A (si veda ancora il citato testo di Nilsson).

C'è ancora qualcosa a proposito della differenza tra le ricerche su alberi e su grafi. Il costo di un nodo ancora da estendere, $\hat{g}(n)$, in un albero è uguale al suo costo teorico minimale $g(n)$, perché, per definizione, c'è un unico cammino dalla radice S a qualsiasi altro nodo. Poiché un grafo può contenere più di un percorso da S ad n, il costo di un cammino trovato può non essere quello minimale e perciò dev'essere chiamato $\hat{g}(n)$. Tuttavia, un algoritmo accettabile che non cambi la sua natura durante la ricerca su grafo produrrà solo cammini ottimali verso nodi estesi, cosicché

(1a)

```
9881 REM ------------------LISTATO 1------------------------------
9884 REM
9885 REM ALGORITMO 'OUT-OF-PLACE', ACCETTABILE
9887 REM
9895 REM ---------------------------------------------------------
9900 R1=0
9910 FOR I=1 TO R9: FOR J=1 TO R9
9915 Q=ASC(E$(I,J))
9920 IF Q=46 THEN 9960
9925 IF Q>64 THEN N=Q-55: GOTO 9935
9930 IF Q<=57 THEN N=Q-48
9935 P1=R9*(I-1)+J
9940 REM -P1 E' IL VALORE DEL PEZZO GIUSTO NELLA POSIZIONE I,J
9945 IF N<>P1 THEN R1=R1+1
9960 NEXT J,I
9965 RETURN
```

(1b)

```
9900  valore dello schema (R1) = 0
9910  for ogni riga I
        for ogni colonna J
9915    :  Q = valore ASCII della I, col. J dello schema E$
9920    :  if il pezzo non è "." (Q ≠ 46)
        :  :  traduci il pezzo Q nel valore "vero" N
9935    :  :  Pl = valore del pezzo nella riga I, col. J nella meta
9945    :  :  if pezzo corrente ≠ valore della stessa posizione nella meta
        :  :  :  nuovo valore di schema = vecchio valore + 1
        :  :  :  end if
        :  :  end if
9960    :  end del ciclo for con variabile J
        end del ciclo for con variabile I
9965  return
```

Listato 1: *L'algoritmo "out of place". Il listato 1a dà l'algoritmo in Basic come va implementato, inserito nel programma SEARCH della parte 1; il listato 1b mostra la versione in pseudolinguaggio. In questi listati, ed in quelli seguenti, il valore di ogni pezzo come stringa è sostituito con il corrispondente valore numerico (cioè il pezzo "1" ha valore 1), con le lettere da "A" ad "F" rappresentate dai valori da 10 a 15, rispettivamente.*

$\hat{g}(n) = g(n)$; ci si riferisce formalmente alla condizione che garantisce questo risultato chiamandola *ipotesi di consistenza*. Tutti gli algoritmi accettabili usati in questo articolo soddisfano tale ipotesi.

Alcuni esempi

I metodi esaustivi di ricerca esaminati nella prima parte di questo articolo (breadth-first, depth-first e depth-first limitato) sono tutti accettabili e rappresentano uno dei due estremi dello spettro delle informazioni: non forniscono informazioni euristiche per la soluzione del problema, cioè $\hat{h}(n) = 0$. L'altro estremo, relativo ad informazioni complete ($\hat{h}(n) = h(n)$), pur essendo teoricamente interessante, è in molti casi impossibile da implementare. Analizzeremo due algoritmi accettabili che si situano tra questi due estremi.

Ricordate che stiamo cercando di definire una funzione $\hat{f}(n)$ che è un limite inferiore al numero minimo di tappe da un nodo n alla meta G. Un algoritmo plausibile è il seguente (vedi listato 1): $f(n)$ sia uguale al numero di quadratini che non sono nella posizione che occupano nella meta G. (Nei giochi dell'8 e del 15 usati come esempio, c'è un'unica meta G). La ragione per cui questo è un limite inferiore al costo effettivo del percorso verso la meta è questa: se uno dei quadratini (escluso lo spazio vuoto) è fuori posto, si impiegherà almeno una mossa, se non di più, per metterlo in ordine; così l'$\hat{h}(n)$ generato da questo algoritmo "out of place" sarà sempre minore o uguale del costo di una soluzione $h(n)$.

La tavola 1a mostra gli schemi usati in questo articolo; la tavola 1b illustra i risultati ottenuti applicando a questi schemi gli algoritmi breadth-first e "out of place". Il confronto delle prime sette righe della tavola 1b suggerisce diverse considerazioni. Per prima cosa, il metodo breadth-first è notevolmente meno efficiente dell'algoritmo "out of place"; il computer che ho usato, che dispone di 20K byte di memoria e può considerare 52 nodi prima di andare in overflow, è in grado di completare al massimo uno schema da quattro mosse col primo metodo, mentre col secondo può portare a termine anche alcuni schemi da dodici mosse. In secondo luogo, entrambi gli algoritmi mostrano un incremento approssimativamente lineare in un certo range (livelli 1-3 e 1-4, rispettivamente) del numero di nodi estesi, col rapporto tra i nodi estesi e il numero minimo teorico di nodi da estendere rispettivamente di 3-1 e 1-1.

Inoltre, questo rapporto cresce progressivamente anche fuori del range di linearità degli algoritmi; questo implica che la massima efficienza raggiungibile da ogni algoritmo decresce con la complessità del problema — in altre parole, quando lo schema diventa più involuto, il valore di h calcolato si allontana sempre di più dall'h teorico andando verso il valore zero (nessuna informazione), e il rendimento dell'algoritmo peggiora (cioè si avvicina a quello di una ricerca esaustiva).

Un'osservazione conclusiva è che gli schemi $(n,1)$ sembrano più facili da risolvere degli $(n,3)$. (Gli schemi con l'ultimo indice uguale sono estensioni l'uno dell'altro.) Questa affermazione è avvallata dal confronto dei numeri nella colonna "nodi estesi" della tavola 1b (che è un metro della difficoltà del

(1a)

Riga	Colonna 1	Colonna 3
1	1 2 3 4 5 6 7 . 8	
2	1 2 3 4 . 6 7 5 8	1 2 3 4 5 6 . 7 8
3	1 2 3 . 4 6 7 5 8	1 2 3 . 5 6 4 7 8
4	1 2 3 7 4 6 . 5 8	1 2 3 5 . 6 4 7 8
5	1 2 3 7 4 6 5 . 8	1 2 3 5 7 6 4 . 8
6	1 2 3 7 4 6 5 8 .	1 2 3 5 7 6 4 8 .
7	1 2 3 7 4 . 5 8 6	1 2 3 5 7 . 4 8 6
8	1 2 3 7 . 4 5 8 6	1 2 3 5 . 7 4 8 6
10	. 1 3 7 2 4 5 8 6	1 3 . 5 2 7 4 8 6
12	7 1 3 2 . 4 5 8 6	1 3 7 5 . 2 4 8 6
14	7 1 3 2 8 4 5 6 .	1 3 7 5 8 2 . 4 6
16	7 1 . 2 8 3 5 6 4	. 3 7 1 8 2 5 4 6
18	7 8 1 2 . 3 5 6 4	3 8 7 1 . 2 5 4 6

(1b)

	Breadth-First			"Out-of-Place"		
Schema	Nodi non estesi	Nodi estesi	Totale	Nodi non estesi	Nodi estesi	Totale
(1,1)	3	1	4	3	1	4
(2,1)	7	4	11	5	2	7
(2,3)	4	3	7	3	2	5
(3,1)	9	8	17	6	3	9
(3,3)	10	9	19	4	3	7
(4,1)	12	11	23	6	4	10
(4,3)	16	21	37	6	4	10
(5,1)	*OM*	(29)	*OM*	9	7	16
(5,3)			*OM*	7	6	13
(6,1)			*OM*	12	9	21
(6,3)			*OM*	13	13	26
(7,1)			*OM*	13	10	23
(7,3)			*OM*	17	17	34
(8,1)			*OM*	14	11	25
(8,3)			*OM*	25	26	51
(10,1)			*OM*	14	13	27
(10,3)			*OM*			*OM*
(12,1)			*OM*	20	20	40
(12,3)			*OM*			*OM*

Tavola 1: *Confronto tra gli algoritmi breadth-first e "out of place" relativamente a problemi particolari. Gli schemi nella tavola 1a possono venir identificati da una coppia di numeri: la riga e la colonna in cui si trova lo schema. Il numero di riga corrisponde al numero di mosse verso la soluzione; gli schemi nella stessa colonna sono sottoinsiemi dello stesso problema. (Corrisponde all'elenco di schemi dato nella prima parte dell'articolo.) La tavola 1b propone un confronto tra gli algoritmi breadth-first e "out of place" su problemi scelti. La relazione che lega i nodi estesi a quelli non estesi è: totale = nodi estesi + nodi non estesi. (*OM* significa che è stato superato il limite di 50 nodi. Le parentesi attorno a 29 nella riga (5,1) mostrano che la ricerca breadth-first conduce a un overflow dopo aver esteso 29 nodi.) La capacità dell'algoritmo "out of place" di risolvere problemi più complessi con la stessa quantità di memoria denuncia la minor potenza del metodo breadth-first.*

problema poiché è in relazione col numero di nodi estesi nella ricerca di una soluzione). Si noti ancora che l'aumento non lineare della colonna dei "nodi estesi" è maggiore per gli schemi (n,3). Ciò suggerisce che l'andamento di un algoritmo fuori del range di linearità di cui abbiamo parlato non può essere rappresentato da una funzione non lineare semplice, ma da un insieme di valori che varia sensibilmente a seconda del particolare schema in questione.

Algoritmo di minima distanza

L'algoritmo di minima distanza qui descritto è il più efficente tra quelli con cui ho lavorato — e non sono riuscito a migliorarlo neanche lasciando cadere il vincolo di accettabilità. L'algoritmo (listato 2) può essere descritto così: per ogni pezzo nello schema (escluso il pezzo "."), al valore dell'algoritmo vengono sommati il numero di righe e il numero di colonne di cui il pezzo dista dalla sua posizione finale nella meta (ignorando gli altri pezzi). Ad esempio, se il pezzo "1" è nella riga 2, colonna 3, allora dista $(2-1) + (3-1) = 3$ quadratini dalla posizione finale nella meta (riga 1, colonna 1) e perciò si aggiunge 3 al valore $\hat{f}$ di quello schema. La tavola 2 mostra il valore dello schema (6,1) con questo algoritmo.

Poiché la figura data è per ogni pezzo una stima moderata di quante mosse siano necessarie per metterlo in ordine (sarebbe più alta se si tenesse conto degli altri pezzi),

(2a)

```
9885 REM --------------------LISTATO 2----------------------------
9887 REM
9890 REM ALGORITMO DELLA MINIMA DISTANZA: ACCETTABILE
9893 REM
9899 REM -----------------------------------------------------------
9900 R1=0
9910 FOR I=1 TO R9: FOR J=1 TO R9
9915 Q=ASC(E$(I,J))
9920 IF Q=46 THEN 9960
9925 IF Q>64 THEN N=Q-55: GOTO 9935
9930 IF Q<=57 THEN N=Q-48
9935 I1=INT((N-1)/R9)+1
9940 REM DATO IL QUADRATO N, CON 1<=N<=15 TROVA
9941 REM (I1,J1)=POSIZIONE DI N NELLO SCHEMA RISOLTO
9945 J1=N-R9*(I1-1)
9950 REM H-CAPPELLO E' LA SOMMA DELLE DISTANZE DI OGNI QUADRATO
9951 REM DALLA POSIZIONE NELLA META; IL QUADRATO '.' NON CONTA
9955 R1=R1+ABS(I-I1)+ABS(J-J1)
9960 NEXT J,I
9965 RETURN
```

(2b)

```
9900  valore dello schema (R1) = 0
9910  for ogni riga I
          for ogni colonna J
9915      :   Q = valore ASCII della riga I, col. J dello schema E
9922      :   if il pezzo non è "." (Q ≠ 46)
          :   :   traduci il pezzo Q nel valore "vero" N
9935      :   :   I1= # di riga del pezzo nella meta
9945      :   :   J1= # di colonna del pezzo nella meta
9955      :   :   nuovo valore dello schema = vecchio valore +
                  (differenza dei valori di riga) + (differenza dei valori di colonna)
9960      :   end del ciclo for con variabile J
          end del ciclo for con variabile I
9965  return
```

Listato 2: *L'algoritmo di minima distanza. Il listato 2a dà l'algorigmo in Basic, pronto per essere inserito nel programma SEARCH della parte 1; il listato 2b è la traduzione in pseudolinguaggio.*

(a) Schema	(b) Meta	(c) Mosse mancanti	
1 2 3	1 2 3	pezzi 1, 2, 3, 6, 8 a posto	=0
7 4 6	4 5 6	pezzo 4 a 0 righe, 1 col.	=1
5 8 .	7 8 .	pezzo 5 a 1 riga, 1 col.	=2
		pezzo 7 a 1 riga, 0 col.	=1
		$\hat{f}$, valore dello schema	=4

Tavola 2: *Calcolo dello schema (6,1) con l'algoritmo di minima distanza. Questo algoritmo somma la distanza di ogni pezzo dalla sua posizione finale (la meta) per ottenere una stima sul numero di mosse della soluzione. La colonna (a) è il problema rappresentato dallo schema (6,1); la colonna (b) è la meta; la colonna (c) elenca i contributi di ogni pezzo al numero totale di mosse necessarie per risolvere lo schema (il pezzo ".", rappresentante lo spazio vuoto, non è considerato nel calcolo).*

l'$\hat{f}$ calcolato come somma di questi valori deve rappresentare un limite inferiore sul vero costo f associato ad ogni schema; quindi questo algoritmo di minima distanza è accettabile.

La tavola 3 illustra i risultati ottenuti da questo algoritmo applicato agli schemi della tavola 1, confrontati con i valori forniti dall'algoritmo "out of place". I risultati sono molto migliori di quelli ottenuti con gli altri algoritmi considerati — in effetti è il primo algoritmo utilizzabile in casi reali. Questo, come già il metodo "out of place", è "completo" (sebbene si possa trovare un controesempio), ma si noti che la crescita non lineare nella colonna "nodi estesi" per l'algoritmo di minima distanza è più graduale ed approssima meglio un andamento rettilineo che non nel caso del metodo "out of place".

Sebbene anche nel primo caso ci si allontani dal valore teorico h per avvicinarsi a zero quando aumenta la complessità del problema, ciò avviene meno bruscamente che per l'algoritmo "out of place"; e questo è dovuto alla maggior quantità di informazioni possedute dall'algoritmo di minima distanza, che si riflette nella generazione di un numero minore di nodi sbagliati rispetto alla soluzione di uno schema.

Tuttavia, sotto un certo aspetto, la tavola 3 è fuorviante: i valori della colonna "nodi estesi" per gli schemi di ordine 12, 14 e 16 sono identici per due insiemi di schemi che, come abbiamo visto, non hanno le stesse complessità, il che potrebbe suggerire che l'algoritmo minimizzi in qualche modo l'effetto dispersivo causato dalle diverse complessità di schemi dello stesso ordine, postulate precedentemente. Ma non è così: la soluzione con l'algoritmo di minima distanza di un numero di schemi di ordine 12 scelti casualmente mi ha assicurato che questa tendenza minimizzante in effetti non c'è; i valori della colonna "nodi estesi" per questi schemi sono 12, 13, 14, 14, 18, 20 e

Schema	"Out-of-Place" Nodi estesi	Totale	Minima distanza Nodi estesi	Totale
(1,1)	1	4	1	4
(2,1)	2	7	2	7
(2,3)	2	5	2	5
(3,1)	3	9	3	9
(3,3)	3	7	3	7
(4,1)	4	10	4	10
(4,3)	4	10	4	10
(5,1)	7	16	6	13
(5,3)	6	13	5	12
(6,1)	9	21	7	15
(6,3)	13	26	7	15
(7,1)	10	23	7	15
(7,3)	17	34	7	15
(8,1)	11	25	8	18
(8,3)	26	51	10	21
(10,1)	13	27	10	21
(10,3)		•OM•	12	24
(12,1)	20	40	14	29
(12,3)		•OM•	14	29
(14,1)		•OM•	16	32
(14,3)		•OM•	16	32
(16,1)		•OM•	18	35
(16,3)		•OM•	18	35
(18,1)		•OM•	20	40
(18,3)		•OM•		•OM•

Tavola 3: *Confronto dell'algoritmo "out of place" col metodo di minima distanza per problemi selezionati.*

>20 (quest'ultimo valore è dato dal superamento della disponibilità di memoria del computer).

Nonostante la normale affidabilità dell'algoritmo di minima distanza, esiste un tipo di problema per il quale è praticamente inefficace. Un esempio di tale schema è dato nella figura 2, ed un'analisi dell'incapacità dell'algoritmo di risolverlo ci fornisce lo spunto per la costruzione di un algoritmo accettabile più potente.

Anche se l'algoritmo assegna allo schema un valore quattro, io non sono stato capace di trovare (a mano) una soluzione con meno di sedici mosse, ed i primi cinquanta nodi dell'albero, che il mio calcolatore è riuscito a generare prima di andare in overflow, non hanno mostrato un apprezzabile avvicinamento alla meta. Infatti, dopo aver generato i nodi da 37 a 40, al livello 10 (figura 2), l'algoritmo li lascia perdere per estendere i nodi dei livelli da 2 a 4, il che significa evidentemente che l'algoritmo ha giudicato non promettenti i nodi dei livelli dal 6 al 10. Sebbene io non abbia trovato un algoritmo accettabile che risolva meglio questo schema, sono sicuro che tale metodo dovrà tener conto del numero extra di mosse che generano i pezzi in posizioni "opposte" rispetto a quelle occupate nella meta (in questo caso il "5" e il "6" e poi il "7" e l'"8").

Algoritmi non accettabili: teoria

Si sa relativamente poco sulle prestazioni degli algoritmi non accettabili — cioè di algoritmi per cui il valore $\hat{h}$ non è necessariamente un limite inferiore al costo effettivo h di una soluzione. Questo perché non sono stati trovati aspetti comuni (relativi alle prestazioni degli algoritmi di ricerca di una meta) che possano riguardare gli algoritmi non accettabili come un'unica classe: un certo algoritmo non accettabile, confrontato con un buon metodo accettabile, può comportarsi molto meglio o molto peggio. Si può trovare addirittura un algoritmo non accettabile meno efficiente di un metodo breadth-first "senza informazioni".

Ad ogni modo, due caratteristiche degli algoritmi non accettabili discendono direttamente dal fatto che non verificano le condizioni di accettabilità. La prima è che non è sicuro che trovino una meta; l'altra è che una meta trovata con un algoritmo non accettabile può non essere una meta *ottimale* (cioè può esistere un altro cammino, più breve, verso lo stesso nodo). Questi svantaggi sono importanti ma non decisivi all'interno di un problema reale, perché, per prima cosa, un certo algoritmo non verrà mai usato se non si sa per esperienza che è adatto a risolvere problemi di quel tipo. (Gli algoritmi non accettabili vengono individuati con procedure di prove ed errori, e l'unica misura della loro efficenza è la capacità o meno di produrre soluzioni a problemi di uguale complessità di cui si conoscano già i risultati.) In secondo luogo, la produzione di un nodo ottimale può non essere importante come quella di una meta, sia essa o no ottimale.

(Altri metodi possono essere usati assieme o al posto di algoritmi non accettabili per produrre una meta. Tutti questi metodi rinunciano alla sicurezza di trovare una meta pur di risparmiare sul numero di nodi intermedi da memorizzare. I successori possono essere tolti dalla memoria quando vengono generati o quando essa è stata

```
9890 REM -------------------LISTATO 3-------------------------
9891 REM
9893 REM
9894 REM ALGORITMO NA-I: NON ACCETTABILE
9896 REM ISTRUZIONI: AGGIUNGERE QUESTO AL LISTATO 2
9897 REM L'ALGORITMO DELLA MINIMA DISTANZA
9898 REM
9899 REM ------------------------------------------------------
9961 REM
9962 L1=100-R1
```

Listato 3: *La modifica necessaria per implementare l'algoritmo NA-I (primo algoritmo non accettabile). Questa modifica al programma Basic del listato 2 fornisce false informazioni al programma SEARCH, rendendolo inefficace anche di fronte ai problemi più semplici.*

```
9890 REM -------------------LISTATO 4-------------------------
9891 REM
9892 REM ALGORITMO NA-II: NON ACCETTABILE
9895 REM
9896 REM ISTRUZIONI: AGGIUNGERE QUESTO AL LISTATO 2
9898 REM
9899 REM ------------------------------------------------------
9955 I9=ABS(I-I1): J9=ABS(J-J1): R1=R1+I9+J9
9957 IF I9>0 AND J9>0 THEN R1=R1+1
```

Listato 4: *La modifica necessaria per ottenere l'algoritmo NA-II (secondo algoritmo non accettabile). Inserita nel programma Basic del listato 2, fornisce prestazioni uguali o migliori del metodo di minima distanza. Poiché si tratta di un algoritmo non accettabile, il successo non è garantito.*

riempita; oppure, con un approccio completamente differente, si può sviluppare attraverso l'albero una ricerca depth-first con livello massimo fissato, che memorizzi solo il miglior nodo incontrato fino a quel punto.)

Alcuni esempi

È facile trovare un esempio di algoritmo non accettabile *cattivo*: basta sottrarre il valore calcolato dall'algoritmo di minima distanza (che è un buon algoritmo) da un numero grande arbitrario. Si ottiene così un algoritmo che assegna un valore alto ad un nodo vicino alla meta (cosicché sarà uno degli ultimi ad essere esteso) ed un valore più basso ad ogni nodo che dista di più dalla meta; si veda l'algoritmo *NA-I*, nel listato 3. Questo metodo, di fronte a problemi di ordine 2 o più, riempie lo spazio di memoria di quasi tutti i computer prima di ottenere una soluzione, perché estende un nodo "buono" solo dopo aver esteso tutti quelli sbagliati nell'albero del problema. Dopo cinquanta nodi (il massimo per il mio calcolatore) del problema (2, 1), l'algoritmo era molto più lontano dalla soluzione di quanto non lo fosse all'inizio.

Fig. 2. *Un esempio di problema che non può essere risolto efficacemente con l'algoritmo di minima distanza. Sebbene l'algoritmo dica che bastano quattro mosse per trovare la soluzione, il computer supera la disponibilità di memoria prima di riuscirvi. Il numero nel tondino accanto ad ogni nodo indica l'ordine in cui i nodi sono stati generati; i numeri senza cerchietto sono i valori* $\hat{f}$ *calcolati dall'algoritmo.*

Viceversa, è abbastanza difficile trovare un algoritmo non accettabile *buono* — cioè che si comporti meglio dell'algoritmo di minima distanza. Infatti è necessaria una gran quantità di lavoro in parecchie direzioni per ottenere anche un solo risultato positivo. L'algoritmo chiamato *NA-II* (listato 4) è un tentativo di migliorare quello di minima distanza nella risoluzione di problemi più difficili, aggiungendo 1 al valore originale di ogni nodo che disti almeno una riga *e* una colonna dalla sua posizione nella meta. Solo riguardo a due schemi si ottengono risultati migliori che con l'algoritmo di minima distanza (vedi la tavola 2), ma per gli altri le prestazioni sono uguali, ed in tota-

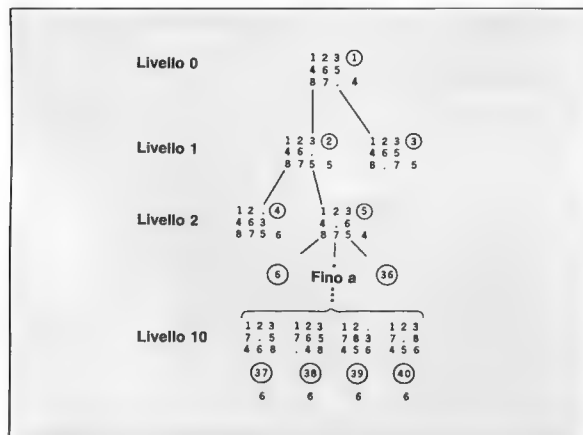
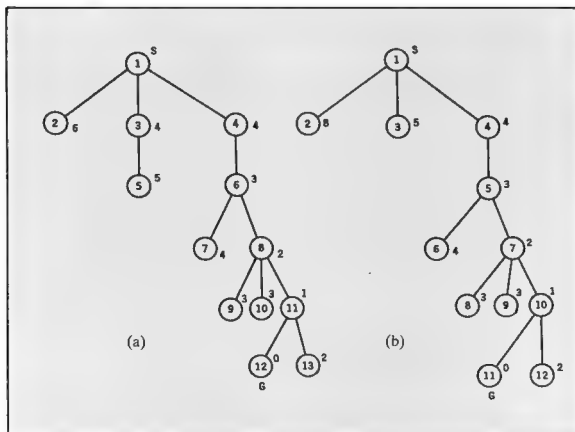



Fig. 3. *L'estensione dello schema (5,1) secondo gli algoritmi di minima distanza (figura 3a) ed NA-II (figura 3b). I numeri all'interno di ogni nodo suggeriscono l'ordine in cui sono stati generati; i numeri all'esterno del cerchietto sono i valori stimati di* f, *calcolati da ogni algoritmo. In questo caso, l'algoritmo non accettabile offre una prestazione migliore.*

le l'algoritmo *NA-II* è dunque migliore.

Le figure 3a e 3b mostrano l'estensione del problema (5,1) con gli algoritmi di minima distanza ed *NA-II*, rispettivamente, l'ordine in cui i nodi si sono estesi ed il valore della stima *h* per ciascun algoritmo. Si noti che l'idea buona dell'algoritmo *NA-II* consiste nel non estendere il nodo 3, mentre il metodo di minima distanza, sottostimando il suo costo (che in realtà è 5), lo ha calcolato invano. Vediamo perciò che *NA-II*, almeno in questo caso, è migliore dell'algoritmo di minima distanza, perché è meno propenso a sottostimare il valore dei nodi (che è una tendenza degli algoritmi accettabili). Inoltre, questo metodo si conferma non accettabile, giacché dà talvolta stime superiori al valore dei nodi (cosa che un algoritmo accettabile non può fare). Il nodo 2 della figura 3b è un esempio di "sovrastima": il suo vero valore è 6, ma la stima dell'algoritmo *NA-II* è 8.

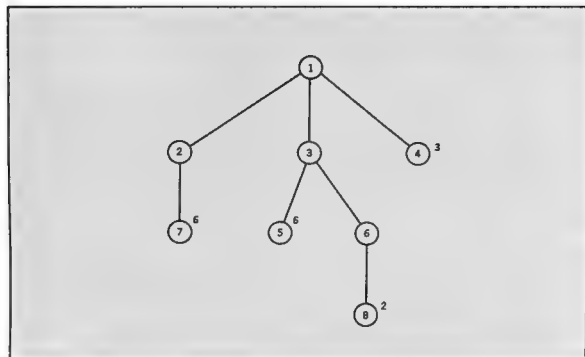
Glossario

Albero del problema: *una rappresentazione grafica dello spazio del problema (o spazio degli stati) che si serve di punti per rappresentare gli stati e di linee tra i punti per la transizione da uno stato a quello seguente; tutti i nodi devono essere generati dal nodo di partenza* S, *lo stato iniziale del problema.*
Algoritmo d'ordinamento: *una formula o una procedura per generare un valore d'ordine che classifichi la probabilità del relativo nodo di essere scelto per l'estensione; il nodo con numero d'ordine minore sarà il primo ad essere esteso.*
Costo (o **valore**): *un valore numerico associato al cammino minimo dal nodo iniziale* S *al nodo in questione* n; *il costo della prima meta che si trova sarà particolarmente importante all'interno del problema da risolvere.*
Estendere: *calcolare tutti i successori del nodo in questione.*
Livello: *il numero di nodi di cui un certo nodo dista da quello iniziale* S.
Meta: *qualsiasi nodo soddisfacente l'insieme di condizioni definite come obiettivo del problema.*
Nodo: *elemento di un albero, rappresentante un certo stato del problema.*
Nodo esteso: *un nodo i cui successori sono già stati calcolati.*
Nodo non esteso: *un nodo che non è ancora stato considerato per essere esteso.*
Rappresentazione dello spazio degli stati: *la riduzione del problema nelle seguenti componenti: le variabili di stato che possono descrivere il problema in tutte le sue possibili configurazioni; gli operatori che generano il prossimo insieme di valori (o stati) per il problema a partire dall'insieme di stati corrente; un nodo iniziale* S; *ed una descrizione (non necessariamente esatta) della meta da trovare.*
Stati*: un insieme specifico di valori per le variabili che definiscono il problema.*
Successori: *i nodi rappresentanti tutti i "prossimi stati" di un nodo (o stato) dato, generati dagli operatori della rappresentazione dello spazio degli stati; il nodo che genera i successori è chiamato padre.*

Osservazioni e discussione

Finora abbiamo usato la parola "costo" solo per riferirci al valore numerico associato al percorso minimo da un nodo alla meta più vicina. Ma il costo di una soluzione può avere altri due importanti significati. Un indice del costo di una soluzione è il numero di nodi estesi dall'algoritmo — è il metro che abbiamo usato per confrontare l'efficienza di due algoritmi. Ma un altro fattore deve essere considerato, oltre alla velocità e alla spesa (espressa in tempo di calcolo). Si tratta della complessità del calcolo della funzione $\hat{h}$. Un algoritmo con più informazioni, pur generando meno nodi, può impiegare per questo calcolo molto più tempo. Se la velocità o la spesa diventano fattori critici prima della quantità di memoria disponibile, è possibile che l'utente decida di usare l'algoritmo con meno informazioni.

– (Domanda 1) Come affronta la ricerca su alberi un algoritmo euri-

stico? (Troverete le soluzioni nel riquadro "Risposte").

– (Domanda 2) Nella descrizione dell'algoritmo di minima distanza lo spazio vuoto non era considerato nella somma delle "distanze dalla posizione giusta". Se invece se ne tenesse conto, l'algoritmo diventerebbe più potente? O meno potente? Sarebbe ancora accettabile?

– Si può avere un albero come quello in figura 4 per un algoritmo accettabile? Sì, ma solo se l'algoritmo sbaglia la stima del valore di uno dei nodi non estesi (4, 5, 7 o 8). Per esempio, se la meta ottimale dista tre nodi dal nodo 4, allora i successori del nodo 8 (o, al più tardi, i *loro* successori) danno ad $\hat{h}$ valori maggiori di 3, cosicché il nodo 4 sarà il primo ad essere esteso. Un algoritmo accettabile raggiungerà sempre prima la meta più vicina.

– Questo stesso albero può riguardare un algoritmo non accettabile? Sì, un algoritmo non accettabile non pone restrizioni alla validità di questo albero.

– (Domanda 3) Ricordate il significato di $\hat{f}$, $\hat{g}$ ed $\hat{h}$: $\hat{f}(n)$ è una stima della distanza dal nodo di partenza S alla meta più vicina, passando per n; $\hat{g}(n)$ e $\hat{h}(n)$ sono stime per le distanze da S ad n e da n alla stessa meta, rispettivamente. Forse che l'albero in figura 4 è una buona ragione per usare $\hat{f}$ al posto di $\hat{h}$ nell'ordinare l'estensione dei nodi?

– (Domanda 4) L'algoritmo di minima distanza si è rivelato il migliore per il gioco dell'8 e del 15. Provate ad aggiungere la riga

9963 RI = RI* F9

alla subroutine per il calcolo di $\hat{h}$ della riga 9900. Si aumenterà così il valore h per i nodi che sono stati precedentemente sottostimati dall'algoritmo accettabile. L'algoritmo diverrà non accettabile, ma sarà anche più "penetrante"? Provare con F9 = 1.01, 1.1, 1.5 (e con altri valori) e sperimentate il nuovo algoritmo con schemi di otto mosse o più.

– (Domanda 5) Perché l'algoritmo non accettabile *NA-I* è peggiore del metodo breadth-first? Un algoritmo di ricerca esaustiva privo di informazioni euristiche non è forse il metodo di ricerca meno efficente?

– Come ho già detto, talvolta è utile affiancare ad un algoritmo non accettabile alcune varianti al metodo di ricerca. In certe situazioni, l'applicazione razionale di uno di questi metodi può essere più efficace, ai fini della ricerca di una meta, dei metodi "puri" descritti nell'articolo.

Fig. 4. *Un'estensione parziale ipotetica ad albero, utilizzata per visualizzare alcune domande del testo.*

Conclusioni

In questo articolo abbiamo trattato la ricerca su alberi e grafi rappresentanti uno spazio degli stati. Altri tipi di alberi (gli alberi AND/OR e gli alberi di strategia nei giochi, per citarne due) si incontrano nelle dimostrazioni di teoremi e nello sviluppo di giochi, e ad essi sono collegati moltissimi problemi.

Ad esempio, come si possono valutare gli algoritmi non accettabili? Quali modifiche richiede la limitatezza di spazio in memoria? Siccome la quantità di intelligenza artificiale che si incontra in un programma richiede almeno il triplo in termini di quantità di lavoro, spero proprio che questo articolo stimoli qualcuno ad approfondire (e diffondere) questa affascinante materia.

Risposte

1. *Un algoritmo euristico affronta la ricerca su alberi sulla base di informazioni sullo stato corrente (cioè sul nodo in questione, senza preoccuparsi della sua relazione con nodi diversi dalla meta) per valutare la possibilità che quel nodo guidi alla soluzione. Ai nodi con più probabilità viene assegnato un valore più basso, cosicché il programma di controllo, scegliendo di estendere il nodo col valore d'ordine più basso, considera proprio quello che probabilmente porterà alla soluzione più breve.*
2. *La variante nel metodo di minima distanza è meno efficace anzitutto perché non è più accettabile. Lo schema*

```
1 2 3
4 5 6
7 . 8
```

è un semplice controesempio. Poiché sia "." che "8" distano un quadratino dalla loro posizione nella meta, l'algoritmo calcolerà il valore 2. Tuttavia, siccome il vero valore della soluzione è 1, basta questo controesempio per dimostrare la non accettabilità dell'algoritmo.
3. *Se l'algoritmo utilizzato è accettabile, l'uso di* $\hat{h}$ *garantisce di trovare la meta più vicina — ciò è matematicamente indiscutibile. Ma se l'algoritmo è non accettabile e, al tempo stesso, relativamente buono, l'uso di*

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$$

può proprio essere una buona idea. Se le stime dei costi $\hat{h}$ *della figura 4 sono esatte una rispetto all'altra, allora*

$$\hat{f}\ (nodo\ 4) = 1 + \hat{h}\ (nodo\ 4) = 1 + 3 = 4 \qquad \hat{f}\ (nodo\ 8) = 3 + \hat{h}\ (nodo\ 8) = 3 + 2 = 5$$

e il nodo 4 verrà perciò esteso prima.
4. *I risultati di questo nuovo algoritmo saranno identici a quelli dell'algoritmo di minima distanza, pur essendo il primo non accettabile. Moltiplicando i risultati per una costante, cambieranno i valori dei nodi, ma non l'ordinamento reciproco. D'altro canto, aggiungendo*

```
9963 IF R1 > F8 THEN R1 = R1 + F9
```

oppure

```
9963 IF R1 < F8 THEN R1 = R1 + F9
```

si modificherà la relazione tra i nodi. Provate anche con altri valori di F8 e F9; suggeriamo 4 come valore iniziale per F8.
5. *Non è vero. Avere informazioni sbagliate è peggio che non averne affatto, ed è proprio il caso di NA-I. Assegnando valori alti a nodi che dovrebbero averli bassi, e viceversa, l'algoritmo fa sì che il programma estenda prima i nodi meno promettenti.*

La compressione dei testi

Come compattare un file, riducendo la quantità di spazio necessaria per memorizzarlo

di *J.L. Peterson*

Un problema sempre presente in qualsiasi sistema è il risparmio di memoria. Non abbiamo mai a disposizione memoria sufficiente per contenere tutte le informazioni che vogliamo. E questo vale sia per i programmi in memoria centrale che per i dati su supporti esterni.

Una semplice soluzione al problema consiste nel comperare più memoria. In particolare nel caso delle memorie esterne a supporti mobili come cassette, floppy disk, nastri magnetici ed anche nastri di carta, questi si possono comperare in quantità dipendente dalle necessità. Il limite alla quantità di memoria disponibile è dato allora da fattori economici.

Un approccio alternativo consiste nello sfruttare meglio i mezzi a disposizione per la memorizzazione. Ecco in cosa può essere utile la *compressione dei testi.* L'idea consiste nel ridurre la quantità di spazio necessaria per memorizzare un file, comprimendolo, cioè rendendolo più breve. La compressione viene effettuata modificando la tecnica di rappresentazione del file. La procedura di *codifica* è realizzata in modo da risultare reversibile; è possibile cioè effettuare una *decodifica* per ottenere il file originale. Ciò è illustrato in figura 1. La speranza è che il file codificato sia più corto di quello originale, e che quindi si risparmi dello spazio.

Il prezzo di questo guadagno di spazio è pagato in termini di tempo di elaborazione. È necessario del tempo addizionale per codificare e decodificare i file quando vengono elaborati. Tuttavia, bisogna notare che i microprocessori sono raramente limitati da questo punto di vista disponendo di solito di cicli d'elaborazione extra.

In effetti il tempo totale d'esecuzione di molti programmi sarà minore per un file compresso, nonostante la sua codifica. Questo perché il tempo dei passaggi di I/O (input/output) per un file compresso è minore dello stesso tempo per un file non compresso, perché ci sono meno bit da leggere o scrivere. Quindi i programmi limitati dall'I/O (come assemblatori e caricatori) possono essere eseguiti più velocemente se i file sono compressi.

Il concetto che sta alla base della compressione dei testi consiste nel trovare il metodo di codifica che richiede il minimo spazio. Sono stati elaborati parecchi algoritmi per la compressione dei testi, e noi ne presenteremo alcuni. In generale questi algoritmi funzionano con qualsiasi tipo di dati: numerici, stringhe di caratteri, ecc. Per gli scopi di questo articolo, comunque, ci limitiamo ai testi, cioè alle stringhe di caratteri. Compresi naturalmente i programmi, la documentazione, le liste di indirizzi, i

La riproduzione di questo articolo è stata concessa da BYTE. Traduzione di Flavio Santini

dati e molti altri file memorizzati all'interno dei calcolatori. Infatti, si possono comprimere anche i programmi oggetto, considerandoli semplicemente come stringhe di byte, ma l'operazione è molto delicata.

La compressione dei testi è realizzata tramite un'attenta analisi della *rappresentazione* dell'informazione nel file compresso. In molti piccoli sistemi, per rappresentare i caratteri si usa il codice ASCII. Il vantaggio principale derivante dall'uso del codice ASCII è che si ottiene una rappresentazione standard e facile da definire. Il maggiore svantaggio è lo spreco di spazio.

L'ASCII è un codice a 7 bit, mentre la maggior parte dei processori trattano byte da 8 bit. Così, 1 bit su 8 (il 12.5%) viene sprecato semplicemente perché si usa un codice a 7 bit in un byte da 8 bit. Inoltre, gran parte dei codici di controllo vengono usati raramente, e molte applicazioni non richiedono l'uso di caratteri sia maiuscoli che minuscoli. Perciò di solito si può facilmente risparmiare un altro bit, ottenendo una riduzione dello spazio occupato di almeno il 25%. Gran parte degli algoritmi presentati qui possono sfruttare questi bit superflui per un risparmio di spazio ancora maggiore.

Si noti tuttavia che questo approccio richiede una descrizione del modo in cui il file compresso deve essere rappresentato. Questa descrizione è data normalmente dalla routine di codifica e decodifica. Il risparmio che si ottiene con la compressione del testo può essere valutato in contrapposizione all'aumento di tempo dovuto alle routine sia di codifica che di decodifica. Inoltre, tipi diversi di file si prestano ad essere codificati meglio con metodi differenti, e quindi possono essere necessarie parecchie routine diverse di codifica e decodifica.

Fig. 1. *Processo di compressione di un testo.*

I trailing blank ed il tab

Un primo semplice passo nella compressione dei file di testi (ma non di codici oggetto) consiste nell'eliminare i blank che si trovano alla fine della riga prima dei caratteri di ritorno del carrello e di capo riga. Questi vengono chiamati *trailing blank*. In sistemi che memorizzano grandi quantità di linguaggio compilato, programmi in Basic o Fortran, gran parte di ogni riga è composta da blank. Tutti i trailing blank possono essere cancellati senza che il significato del file risulti modificato.

Per ridurre il numero di blank in qualsiasi punto di una riga, si può usare il tab. Particolarmente con programmi strutturati a blocchi, come Algol, Pascal o PL/1, o con linguaggi orientati a colonna, come il Fortran o il linguaggio assembler, il tab può essere molto utile nella compressione dei testi. Possono essere usate tecniche di tabulazione di due tipi. Una è chiamata *tabulazione fissa*. In questo caso, la tabulazione è effettuata ogni n colonne, dove n è una costante fissa del sistema. Di solito $n = 8$, sebbene alcuni studi abbiamo mostrato che $n = 4$ ed $n = 5$ dovrebbero dar luogo ad un ulteriore risparmio. L'altra possibilità consiste nell'utilizzare una *tabulazione variabile*. In questo caso, per ogni file vengono fissate le lunghezze di tabulazione. Ciò richiede una valutazione di quali siano le tabulazioni migliori (cioè quelle che producono la miglior compressione). Inoltre è necessario specificare che tipo di tabulazione dev'essere usato per ogni file. Ciò si può fare semplicemente inserendo un elenco delle tabulazioni possibili all'inizio di ogni file.

Questo dizionario dovrebbe servire per inizializzare le tavole per la routine di decodifica che dovrebbe rimpiazzare ogni tab con un numero appropriato di blank. In questo modo si possono usare tabulazioni diverse per linguaggi di programmazione o insiemi di dati diversi.

Caratteri multipli

La sostituzione dei trailing blank con tab serve per comprimere stringhe di caratteri blank multipli. In alcuni casi ci si può trovare di fronte alla ricorrenza frequente di stringhe di caratteri multipli diversi dal blank. Per esempio, le tecniche di disegno col computer richiedono spesso la memorizzazione di lunghe sequenze di caratteri identici, come nel caso della figura 2. L'idea è ora quella di sostituire una stringa di n caratteri identici con il numero n ed un carattere, risparmiando così $n - 2$ caratteri. Il numero può essere rappresentato in un byte. Se supera 256, può essere espresso come 256, seguito dal carattere e da un altro numero e un altro carattere per il resto.

La codifica consiste nel contare i caratteri identici finché non se ne trova uno diverso, per poi avere in

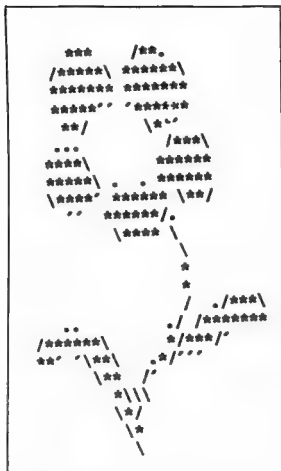



Fig. 2. *Un file che può essere compresso con tecniche particolarmente semplici. Il file originale è un'immagine per video di 24 per 80, cioè 1920, caratteri. Eliminando i trailing blank ed usando la tabulazione da 8 colonne, riduciamo la dimensione del file a 412 caratteri, con un risparmio del 78.6%.*

output il numero ed il carattere. La decodifica si occupa semplicemente di espandere il numero ed il carattere nella sequenza relativa.

Ovviamente, perché la tecnica abbia successo *n* dev'essere maggiore di 2. Se *n* fosse 1, questo metodo in effetti raddoppierebbe la dimensione del file. Poiché questo è normale nei file di testi, in questi casi bisogna usare un approccio più sofisticato.

Vogliamo sostituire con un numero ed un carattere le sequenze di caratteri identici, tralasciando però i caratteri singoli o doppi. C'è il problema di rappresentare i caratteri multipli in modo che il numero non venga interpretato come un carattere. Una soluzione comune è data dall'uso di una sequenza *escape*, che è un mezzo per indicare che i caratteri che seguono devono essere interpretati in un modo particolare. Per creare una sequenza escape, si sceglie un carattere usato raramente (o meglio mai). Ad esempio, in ASCII si può usare uno dei codici di controllo o un carattere speciale. L'ASCII dispone anche di un carattere escape, ma nel caso che venga già usato per altri scopi, se ne può usare un altro. Dunque una sequenza di *n* caratteri identici sarà rappresentata dal carattere escape, il valore *n* ed il carattere ripetuto. La figura 3 mostra il testo della figura 2 compresso con questa tecnica.

Ciò consente di rappresentare come al solito un testo normale, fatta eccezione per il carattere escape. Il problema che ci si presenta ora è quello di saper rappresentare il carattere escape, se è presente nel testo di input (non ancora compresso). Se lo copiamo brutalmente nel file compresso, il decodificatore penserà (a torto) che si tratti dell'inizio di una sequenza escape ed interpreterà i due caratteri seguenti come se rappresentassero una sequenza di caratteri identici (è lo stesso problema che i progettisti di linguaggi trovano per la rappresentazione tra apici di una stringa contenente altri apici). Ci sono molti modi per risolvere questo problema: eliminare dal testo tutti i caratteri escape; rimpiazzare tutti i caratteri escape con una sequenza escape speciale, per esempio con un contatore 0; sostituire tutti i caratteri escape con un carattere escape, il numero 1 e un altro carattere escape, considerandoli come sequenze di lunghezza 1.

Uno qualsiasi di questi metodi consentità di codificare e decodificare facilmente e correttamente qualsiasi file.

Si noti che, per quanto riguarda la scelta del carattere escape, possiamo usare sempre lo stesso o sceglierne uno diverso per ogni file. Se facciamo in quest'ultimo modo, dobbiamo scorrere preliminarmente il file per cercare un carattere che non venga mai usato. Poi lo inseriamo all'inizio del file per permettere all'algoritmo di decodifica di riconoscere qual è il carattere escape usato.

Sostituzione delle parole chiave

Un tipo molto comune di file memorizzato nei computer è il file di un programma. A causa della loro forma stilizzata e della loro sintassi, i programmi si prestano particolarmente ad essere compressi. Intanto la memoria richiesta può essere ridotta sensibilmente con la tecnica di sostituzione dei trailing blank con il tab. Ma un ulteriore importante risparmio deriva dalla sostituzione delle parole chiave.

Quasi tutti i linguaggi di programmazione si servono di *parole chiave* o *parole riservate*: esempi in Fortran sono INTEGER, FORMAT, CALL; in Basic invece LET, READ, PRINT, REM e così via. Essendo usate più volte nel programma, queste parole sono le

```
$32 $3*$4 /**.@$30 /$5* \  $6* \ @$30 $7* $7*@$30 $5* '  $6*@$31 **/$6  \ ** @$30
$3 $9  /$3* \ @$29  $4* \ $7  $6*@$29  $5* \        $6*@$29   \ $4*   $6*    \ **/@$31  "
$6*/.@$36  \ $4*   \ @$43   \ @$43  *@$43  *@$43  /    /$3* \ @$30    $9   /  /$7* \ @$27
/$6* \ $6*/$3*/'@$26/**'   '\ ** \ $3  */$3'@$26'$6 \ **/'@$35*$3 \ @$35 \ */@$36 \
*@$37 \ @$37 \ @
```

Fig. 3. *Ulteriore compressione del file mostrato in figura 2, effettuata rimpiazzando i caratteri identici multipli con una sequenza escape. Questa nel nostro caso è data dal carattere $ seguito dal numero di ripetizioni e dal carattere che deve essere ripetuto. Questo metodo è vantaggioso solo quando il numero di ripetizioni è maggiore di 2. Il numero normalmente è contenuto in un byte, ma qui è rappresentato in decimale. Il carattere @ rappresenta il ritorno del carrello ed il passaggio a capo riga. Il file di figura 2 viene rappresentato in soli 287 caratteri. Così si riduce il file al 14.9% della sua dimensione originale.*

prime candidate ad essere compresse.

Due sono i metodi più usati. Il primo consiste nel sostituire ogni parola chiave con una sequenza escape. Questa potrebbe essere data dal codice escape, seguito da un numero che indichi qual è la parola chiave in questione. Questo metodo offre il vantaggio di poter trattare tante parole chiave quanti sono i numeri rappresentabili in 1 byte, e può essere particolarmente utile per codici operativi simbolici di linguaggi assembler.

Una tecnica alternativa consiste nel cercare tra i codici esistenti quelli che non vengono mai usati. Ad esempio, se si usa il codice ASCII, molti caratteri di controllo, alcuni caratteri speciali ed in certi casi anche i caratteri minuscoli non vengono solitamente usati. Se si usa l'ASCII a 7 bit con byte a 8 bit, allora il bit rimanente può servire per definire 128 nuovi codici inutilizzati. Questi vengono associati alle parole riservate più comuni. Uno di essi corrisponderà ad un carattere escape, per il caso in cui uno dei codici che non si pensava fossero usati compaia invece nel file in input. In fase di codifica, vengono identificate le parole riservate presenti nel file in input e sostituite con il codice speciale relativo, come mostra la figura 4. Se si incontra uno dei codici speciali, lo si rappresenta con una sequenza a 2 caratteri costituita dal codice escape seguito dal carattere in input. Per quanto riguarda invece la decodifica, tutti i codici speciali sono rimpiazzati dalle corrispondenti parole chiave, tranne quelli preceduti dal codice escape, che vengono copiati direttamente in emissione.

A questo punto emerge una questione. In ogni particolare linguaggio c'è un numero fisso, e di solito abbastanza piccolo, di parole riservate, ma queste variano da linguaggio a linguaggio. Di conseguenza può variare di molto la corrispondenza tra i codici speciali e le parole chiave. Nei sistemi con un unico linguaggio (per esempio in quelli che offrono solo il Basic) questo problema non si pone, ma negli altri casi lo si deve prendere in considerazione.

```
 10 READ A                  10 $5 A
 20 IF A=0 THEN 110         20 $2 A=0 $6 110
 30 IF A>0 THEN 80          30 $2 A>0 $6 80
 40 LET B=-A                40 $3 B=-A
 50 LET R=SQR(B)            50 $3 R=SQR(B)
 60 PRINT A,R," $ "         60 $4 A,R," $$ "
 70 GO TO 10                70 $1 10
 80 LET R=SQR(A)            80 $3 R=SQR(A)
 90 PRINT A,R               90 $4 A,R
100 GO TO 10               100 $1 10
110 END                    110 $7
```

Fig. 4. *La compressione di un programma Basic con la sostituzione delle parole chiave. Le parole chiave (1)* GO TO, *(2)* IF, *(3)* LET, *(4)* PRINT, *(5)* READ, *(6)* THEN *e (7)* END *sono rimpiazzate da una sequenza escape formata dal carattere escape $ seguito dal numero associato alla parola chiave. Si noti che il carattere escape era presente anche nel programma originale, e quindi è stato rappresentato tramite una speciale sequenza escape $$. In effetti, tutti i blank che delimitano le parole chiave vengono inclusi in essa. Così la riga 10 diventa 10$5A, e $5 rappresenta* "READ".

Sono possibili più soluzioni. Intanto si possono usare ruotine di codifica e decodifica separate per ogni linguaggio, lasciando al programmatore libertà di scelta. In secondo luogo si può contrassegnare ogni file compresso con un byte che indichi se è un file in Basic, in Fortran, o nel linguaggio X. Allora si è in grado di dire al codificatore in che modo il file dev'essere codificato, oppure è lui stesso che riconosce (o calcola) se si tratta di un file in Fortran, Basic o linguaggio X ed applica il giusto algoritmo di compressione. Il file compresso viene contrassegnato a seconda della codifica. Il decodificatore legge il contrassegno ed applica il conseguente piano di decodifica.

Un terzo approccio è più generale, ma teoricamente più dispendioso. Ciò che varia negli algoritmi di codifica e decodifica per i diversi tipi di file è semplicemente la tavola delle relazioni tra parole chiave e codici carattere. Quindi, un altro metodo consiste nel premettere ad ogni file compresso un dizionario degli abbinamenti tra codici carattere e parole chiave. Questo elenco chiarisce il significato dei codici carattere speciali indicando le parole chiave ad essi corrispondenti.

Abbreviazione delle sottostringhe

L'idea di premettere un dizionario delle abbreviazioni all'inizio di un file compresso suggerisce di estendere il metodo di sostituzione delle parole chiave anche a file il tipo più generico. Il concetto è piuttosto semplice. Si tratta di selezionare le sequenze di caratteri che ricorrono più frequentemente in un file e di sostituirle con codici di caratteri speciali. Per consentire la decodifica, si inserisce all'inizio di ogni file un elenco dei codici corrispondenti alle stringhe di caratteri rimpiazzate. Questo metodo di compressione dei testi è molto valido, specialmente per i programmi o i linguaggi naturali, giacché le parole chiave, i nomi di variabile o altre parole (come *the* e *and* in inglese) ricorrono molto spesso.

Ma anche in questo caso sorgono alcuni problemi. Il più consistente è quello della scelta delle stringhe di caratteri da abbreviare. Nel caso di programmi scritti in un linguaggio particolare, le parole chiave si ripetono più volte e quindi è chiaro che conviene sostituirle, ma quali sono le stringhe di caratteri che, in generale, è bene sostituire? Le possiamo individuare solo esaminando il file, poiché non ci sono stringhe preferibili in generale.

L'obiettivo, naturalmente, è quello di ottenere il massimo ri-

sparmio di spazio. In questo senso siamo limitati dal numero di codici disponibili per le sostituzioni. Se usiamo i codici inutilizzati, siamo, di solito, ristretti ad un numero che varia tra i 10 ed i 50 codici. Se estendiamo l'insieme di caratteri (per esempio usando i codici a 8 bit con l'ASCII a 7 bit) allora possiamo disporre di almeno 128 codici. L'uso di una sequenza escape può fornirne fino a 256, ma al prezzo di almeno 2 caratteri per abbreviazione. In tutti i casi, il numero di codici disponibili sarà sempre limitato, diciamo ad *m*. Così dobbiamo scegliere per l'abbreviazione le *m* stringhe che consentono il maggior risparmio di spazio.

Non è detto che si scelgano sempre le *m* stringhe che compaiono più frequentemente. Consideriamo le due stringhe *per* e *compressione dei testi*. Se *per* compare 100 volte e *compressione dei testi* solo 15 volte, quale dobbiamo sostituire? Sostituendo *per*, sequenza di 3 caratteri, con un singolo codice d'abbreviazione, risparmiamo 2 caratteri (supponendo il codice di abbreviazione di 1 byte) per ogni volta che la stringa compare nel testo, cioè in totale 200 caratteri. Sostituendo *compressione dei testi*, sequenza di 22 caratteri, si risparmiano 21 caratteri alla volta, cioè 315 caratteri. Dunque, in generale, dobbiamo rimpiazzare la sequenza di caratteri che ha massimo prodotto tra lunghezza e frequenza. Un esempio di sostituzione di sottostringhe è illustrato dalla figura 5.

Il problema della codifica diventa quello di trovare le *m* sequenze con massimo prodotto lunghezza-frequenza, per rimpiazzarle tutte con gli *m* codici di abbreviazione, dopo aver premesso l'elenco delle abbreviazioni all'inizio del file compresso. In fase di decodifica invece il problema si riduce semplicemente a leggere il dizionario ed a sostituire tutti i codici con le corrispondenti stringhe di caratteri.

Il codice di Huffman

Tutti i metodi di compressione dei testi discussi finora sono simili, nel senso che si riducono a lavorare con un dato codice di caratteri ed una data struttura di byte. Un ulteriore risparmio si può ottenere ricodificando la rappresentazione stessa dei codici carattere. Quasi tutte le rappresentazioni dei codici carattere utilizzano una dimensione fissa: 4 bit per un decimale codificato in binario (BCD), 7 bit in ASCII e 8 bit in EBCDIC. In questo modo si spreca molto spazio.

Consideriamo il semplice problema di codificare i 4 caratteri A, B, C e D. Se usiamo una dimensione fissa per codice, allora abbiamo bisogno di 2 bit per ogni carattere, così:

A	00
B	01
C	10
D	11

Ma supponete che la lettera A ricorra nel testo al 50%, B al 25% e C e D si dividano equamente il rimanente 25%. Allora il seguente codice carattere a lunghezza *variabile* produrrà un testo di lunghezza *media* minore.

A	0
B	10
C	110
D	111

Per calcolare la lunghezza media del testo, si consideri che, su *n* caratteri, *n*/2 saranno A e richiederanno 1 bit ciascuno, *n*/4 saranno B e vorranno 2 bit ciascuno ed i rimanenti *n*/4 saranno C o D a 3 bit ciascuno. Così il numero totale di bit per rappresentare *n* caratteri è:

$$1\,(n/2) + 2\,(n/4) + 3\,(n/4) = 1.75\,n$$

Confrontando questo risultato con i 2*n* bit richiesti dal codice a lunghezza fissa, vediamo che abbiamo risparmiato il 12.5% della dimensione totale del file.

Fig. 5. *La compressione dei testi per sostituzione di sottostringhe. Queste vengono rimpiazzate da codici d'abbreviazione (qui abbiamo usato sequenze escape). All'inizio del file si inserisce un dizionario dei significati delle abbreviazioni.*

Dizionario:
$A "per"
$B "compressione dei testi"
$C "computer"

Testo:
Questo articolo ha $A argomento la $B nel $C, $A consentire una maggior disponibilità di memoria nel $C stesso. ...

Codificare e decodificare con lunghezze variabili è un po' più difficile che non con lunghezze fisse, ma non eccessivamente. Bisogna solo lavorare molto di più sui bit. Per codificare una stringa come ABAABCDAB, concateniamo semplicemente le rappresentazioni a bit di ogni carattere, superando, se necessario, i limiti dei byte.

A	B	A	A	B	C	D	A	B
0	10	0	0	10	110	111	0	10

Per la decodifica, dobbiamo leggere da sinistra a destra, esaminando ogni bit. Per la stringa 01001100, notiamo che il primo bit è uno 0. Poiché solo A comincia con 0, il primo carattere è A. Il secondo bit è un 1, ma guardando al bit seguente notiamo che il carattere corrispondente deve essere B. Spostiamo i due bit di B, e continuiamo. Il bit seguente è 0, quindi il carattere è A. Poi c'è un 1, e dunque si ha B, C o D. Poi ancora un 1, quindi B è escluso. Il bit che segue indica finalmente C. L'ultimo carattere è poi A. Così il testo decodificato risulta ABACA.

I file di testi memorizzati nel computer possono avvalersi proficuamente del codice di Huffman. Lo si può usare ogniqualvolta le probabilità dei codici carattere non sono uguali. In effetti, più le probabilità sono diverse, migliori sono i risultati della compressione con il codice di Huffman. Guardando una tabella delle frequenze relative alle lettere nella lingua italiana, notiamo che hanno valori molto diversi, e quindi si prestano bene ad

essere compresse con il codice di Huffman.

Per costruire un codice di Huffman, si usa un algoritmo molto semplice (ci riferiamo alla figura 6). Per prima cosa è necessario calcolare le probabilità relative ai caratteri da codificare. Per questo bisogna effettuare un passaggio attraverso un testo campione, un pezzo di file, l'unione di parecchi file, a scelta, contando la frequenza dei diversi caratteri. Poi bisogna ordinare i caratteri secondo la loro frequenza. Prendiamo i due caratteri meno frequenti, e li combiniamo in un *supercarattere*, la cui frequenza è la somma di quelle dei due caratteri. Il codice relativo ai due caratteri sarà il codice del *supercarattere* seguito da 0 per un carattere e da 1 per l'altro. Ora eliminiamo dalla lista i due caratteri usati meno frequentemente ed inseriamo in essa il nuovo *supercarattere* nel posto corrispondente alla sua frequenza. Il processo continua finché tutti i caratteri ed il *supercarattere* sono ridotti da un solo *supercarattere*. Quel che si ottiene è un codice di Huffman di lunghezza media di codice minima. Il codice di Huffman può essere rappresentato meglio tramite un albero binario con i nodi terminali (le foglie) corrispondenti ai caratteri codificati.

Il codice di Huffman può dare buoni risultati nella compressione dei testi, in certi casi riducendo la dimensione di un file di più della metà. La tecnica di base può essere migliorata con alcuni accorgimenti. Ad esempio, si possono usare coppie di caratteri, invece di caratteri singoli, come base di codifica. Ciò richiede una tabella di frequenze più grande, perché ora dobbiamo calcolare le frequenze delle coppie, ed anche una tabella più estesa di associazioni tra coppie di caratteri e codici di Huffman, ma consente di ottenere un risparmio notevole.

Un'altra possibilità consiste nell'uso del codice di Huffman condizionale. L'idea è quella di utilizzare il fatto che la probabilità (la frequenza) di un carattere varia a seconda di quale è il carattere che lo precede. Per esempio, la probabilità di trovare una Q seguita da una U è vicina a 1, mentre quella di avere U seguita da U è praticamente nulla. Perciò una codifica ottimale dovrebbe assegnare ad una U che segue Q un codice molto breve ed ad una U che segue un'altra U un codice molto lungo. L'algoritmo di codifica comprende il calcolo della frequenza con cui ogni carattere segue tutti gli altri. Si deve quindi calcolare un codice di Huffman separato per i caratteri successivi ad ogni carattere. Il piano di codifica tiene conto dell'ultimo carattere codificato e se ne serve per scegliere il codice da assegnare al carattere seguente. Anche l'algoritmo di decodifica deve memorizzare l'ultimo carattere decodificato per essere in grado di scegliere il giusto metodo di decodifica. I codici di Huffman sono vera-

Fig. 6. *Il codice di Huffman per le lettere dell'alfabeto inglese, basato sulle loro probabilità (frequenze). La lunghezza dei codici è inversamente proporzionale alla frequenza con cui le relative lettere compaiono nella lingua inglese (più o meno come nel codice Morse). Le lunghezze dei codici variano da 9 bit (per la z e la j) a 3 bit (per la e e la t). La lunghezza media è di 4.1885 bit per lettera. Con un codice a lunghezza fissa ne sarebbero necessari 5, ed il risparmio è del 16%.*

mente molto semplici, ma possono essere resi più complessi se si vuol ottenere una compressione migliore. Tuttavia, anche usando codici di Huffman semplici, possono sorgere dei problemi. Intanto si noti che la codifica e la decodifica di Huffman comportano comunque un grosso lavoro sui bit, che può dar luogo a programmi molto lenti. Poi, la compressione migliore si ottiene con il codice di Huffman se le frequenze dei caratteri di un file sono molto diverse tra loro, e questa caratteristica varia da un file all'altro. Perciò è meglio effettuare codifiche diverse per ogni file. Ciò si può fare premettendo ad ogni file il relativo codice (come si fa con i dizionari delle abbreviazioni), anche se così si aumenta la dimensione del file (in misura considerevole per i file piccoli).

Inoltre, la caratteristica della lunghezza variabile dei codici di Huffman li rende estremamente soggetti ad errori di trasmissione e memorizzazione. In un codice a lunghezza fissa, modificando un bit, si varia solo il carattere corrispondente, mentre col codice di Huffman, a causa di un errore sulla lunghezza ipotetica del carattere sbagliato, possono venire decodificati in modo scorretto sia quel carattere che tutti i successivi. (Un problema simile si avrebbe in un codice a lunghezza fissa se si aggiungesse o togliesse un bit.) Così, per sicurezza, è meglio inserire nel file controlli d'errore in abbondanza, anche a costo di aumentarne la dimensione.

In alcuni settori i codici di Huffman si rivelano di grande utilità. Pensate ad un sistema *word processing* che memorizza file su una periferica seriale a bassa velocità, per esempio su cassette. Poiché il sistema ha uno scopo particolare, si possono calcolare le frequenze attese per tutti i caratteri della lingua italiana, con un unico codice di Huffman per tutti i file. La codifica e la decodifica possono essere effettuate automaticamente dalle routine di controllo del nastro. Oppure entrambe potrebbero essere inglobate nell'hardware stesso della periferica come una logica speciale o un piccolo processore con una tavola di codifica/decodifica su ROM. Questo metodo di codifica/decodifica risulterebbe totalmente trasparente all'utente. L'unico effetto prodotto sarebbe la capacità di memorizzare un numero grande ma variabile di "caratteri" su di una quantità fissa di nastro.

Conclusioni

La quantità di spazio disponibile per memorizzare informazioni può essere ridotta sensibilmente con semplici tecniche di compressione dei testi, come quelle che abbiamo introdotto in questo articolo. Ognuna di esse offre un risparmio di spazio nella memorizzazione di molti file. E molte tecniche possono essere usate una dopo l'altra per ottenere una compressione a più stadi. La compressione dei testi può essere un metodo semplice ed efficace per aumentare la quantità di memoria disponibile al prezzo di pochi cicli d'elaborazione.

UN TANDY PER AMICO.
COLOR COMPUTER TRS 80/16 K
L. 750.000 + IVA
Il grande personal computer capace di essere tutto: un vero e proprio gestionale, un video-gioco intelligente con le cartucce più sofisticate, un potente elaboratore di dati programmabile ed espandibile, un avanzato sistema computer-grafico a colori.
Soprattutto un amico.
1150
1050
1200
150
BUSTOUT
REBIT COMPUTER
A DIVISION OF G.B.C
Tandy

RIVISTE JACKSON.
LA VOCE PIÙ AUTOREVOLE NEL CAMPO DELL'ELETTRONICA E DELL'INFORMATICA.

TUTTA L'INFORMATICA

	INFORMATICA OGGI	COMPUTERWORLD ITALIA	BIT	PERSONAL SOFTWARE	L'ELETTRONICA
Hardware					
Unità centrali	●	●			
Super computer	●	●			●
Main frame	●	●			●
Supermini a 32 bit	●				●
Minicomputer a 16 bit	●				●
Microprocessori a 16/32 bit			●		●
Microprocessori a 8 bit			●	●	●
Microcomputer single-chip					●
Micro bit-slice					●
Memorie centrali					
Memorie cache bipolari	●				
RAM statiche			●		●
RAM dinamiche			●		●
ROM			●	●	●
EPROM			●	●	●
Memorie di massa					
Unità a dischi rimovibili	●	●	●		
Unità a dischi Winchester da 14 inches	●	●	●		
Unità a dischi Winchester da 8 inches	●	●	●		
Unità a dischi Winchester da 5, 25 inches	●	●	●		
Unità a floppy	●		●	●	
Unità a minifloppy	●	●	●	●	
Unità a microfloppy			●	●	
Unità a nastro da 1/2 inches	●	●			
Unità a nastro da 1/4 inches	●	●			
Unità video tape di backup	●		●		
Mass Storage Systems	●	●			
Dischi ottici	●	●			●
Dischi a stato solido	●	●			●
Memorie a bolle	●				●
Memorie EEPROM	●				●
Terminali					
Unità CRT alfanumeriche	●	●	●		
Unità grafiche storage	●	●	●		
Unità grafiche refresh	●		●		
Unità grafiche raster-scan	●	●	●		
Unità a colori	●		●		●
Unità a plasma			●		
Display video a LED			●		●
Display video a cristalli liquidi			●		●
Stampanti					
Unità seriali ad aghi ad impatto	●	●	●		
Unità seriali a margherita	●	●	●		
Unità parallele	●		●		
Unità elettrostatiche	●		●		●
Unità termografiche	●		●		
Unità a Laser	●	●			●

	INFORMATICA OGGI	COMPUTERWORLD ITALIA	BIT	PERSONAL SOFTWARE	L'ELETTRONICA
Apparecchiature per trasmissione dati:					
Accoppiatori acustici	●		●		●
Modem	●				●
Modem eliminator	●				●
Multiplexer	●				●
Concentratori	●	●			●
Controllori di linea	●	●			●
Communication Processor	●	●			●
Sottosistemi X.25	●	●			●
Sottosistemi SNA	●	●			●
Reti locali					
Broadband	●	●	●		●
Baseband	●	●	●		●
Schemi a polling	●	●			●
Schemi CSMA/CM	●	●	●		●
Schemi token passing	●	●	●		●
Cavi coassiali	●	●	●		●
Fibre ottiche	●	●	●		●
Computer graphics					
Sistemi integrati	●	●	●	●	●
Terminali grafici intelligenti	●	●	●		●
Plotter a penna	●		●		●
Plotter elettrostatici	●		●		●
Digitalizzatori	●	●	●	●	●
Joystick	●	●	●	●	
Mouse	●	●	●		●
Sistemi industriali					
Controllori programmabili	●				●
Unità CAD/CAM	●	●	●		●
Sensori controllori di processi	●				●

JACKSON.

...la sicurezza di scegliere il meglio.

Interfacce A/D e D/A					●
Robot	●	●	●		●
Software					
Sistemi operativi					
Single user	●		●	●	●
Real-time	●	●	●		●
Multi-tasking	●	●	●	●	●
Multi-processor	●	●	●	●	●
Time-sharing	●	●			●
A memoria virtuale	●	●			
Con intelligenza artificiale	●	●	●		●
Linguaggi					
Assembly			●	●	
Interpreti	●		●	●	●
Compilatori			●	●	
Linguaggi procedurali	●	●	●	●	
Linguaggi strutturati	●	●	●	●	
Linguaggi di Query	●	●	●	●	
Linguaggi non procedurali	●	●	●	●	
Linguaggi "naturali"	●	●	●	●	
File management e data base					
Indexing	●	●	●		
Ashing	●	●	●	●	
Data Base gerarchici	●	●	●	●	
Data Base reticolari	●	●	●	●	
Data Base relazionali	●	●	●	●	

Software tool					
Screen Editor	●	●	●		
Form editor	●	●	●		
Debugger simbolici	●	●	●		
Query	●	●	●		
Report writer	●	●	●		
Application generator	●	●	●		
Protocolli di comunicazione					
Asincroni	●	●	●		●
Sincroni	●	●	●		●
BSC	●	●	●		
SDLC	●	●			●
HDLC	●	●			●
SNA	●	●			●
X 25	●	●			●
Bus contention	●	●			●
Srumenti applicativi					
Word Processing	●	●	●	●	●
Posta elettronica	●	●	●		●
Simulatori finanziari	●	●			●
Gestori di agenda	●	●			●
Package grafici	●	●	●	●	●
Package scientifici	●	●	●		●
Package amministrativi	●	●	●	●	●
Package industriali	●	●	●	●	●

elektor
La rivista di elettronica, scienza, tecnica e diletto.
Dal 1983 anche **Elektor**, il più prestigioso mensile europeo di elettronica per appassionati, tecnici e hobbisti, è pubblicato in Italia dal Gruppo Editoriale Jackson.
Pubblicato in quasi tutte le principali lingue europee (inglese, francese, olandese, tedesco, spagnolo, greco e italiano), Elektor ha acquisito un successo che si può ragionevolmente basare su quasi 500.000 copie in tutta Europa.
La prima edizione, uscita nei Paesi Bassi, nacque 23 anni fa: un successo che non ha precedenti in Europa, in questa fascia di lettori.
Ciò che contraddistingue Elektor è l'approccio alla materia: semplice, ma non per questo meno approfondito; di estremo interesse sia per lo sperimentatore sia per il professionista di elettronica.
Proponendo realizzazioni e sperimentazioni pratiche, tutte originali, con l'impiego dei più moderni componenti discreti e integrati, **Elektor** stimola i propri lettori a seguire da vicino ogni progresso in elettronica, con tutte le migliori indicazioni tecniche e divulgative di supporto.
Gli speciali dell'anno
Elektor non è solo una rivista: è tutta una serie di pubblicazioni al contorno, che riassumono le migliori applicazioni europee di elettronica, con tutta una serie di progetti collaudati da una redazione internazionale.
Un numero speciale, edito annualmente, presenta i 100 progetti migliori: tutti garantiti e testati.
Dal 1983 **Elektor** è Jackson: un'ulteriore garanzia di professionalità.
elektor
elettronica - scienza tecnica e diletto
n° 40
Settembre 1982
L. 2.500
72
76
riconoscitore di parole e trigger ritardato

INFORMATICA OGGI
Rivista di elaborazione dati e telematica.
In soli due anni, **Informatica Oggi** è divenuta il punto di riferimento obbligato per chiunque, in Italia, desideri un mensile approfondito e aggiornato di informatica professionale.
Informatica Oggi è la rivista dei sistemi, dai microcomputer della fascia alta ai mini, ai supermini, ai mainframe.
Terminali, periferiche, sistemi di trasmissione dati e telematica costituiscono il fondamentale complemento di informazione tecnica che la rivista fornisce.
Strutturatasi di recente con una redazione fissa nella Silicon Valley, in California, oltre alla redazione di Milano, **Informatica Oggi** è riconosciuta come la rivista di informatica professionale più seria e completa non solo in Italia, ma certamente anche in Europa. Da gennaio 1982 è pubblicata anche in edizione spagnola.
Informatica Oggi: il meglio del mercato editoriale informatico in Italia, con il necessario corredo di informazioni economiche, su nuovi prodotti, di interviste, notizie flash, rapporti sulle più importanti manifestazioni fieristiche e congressuali, in Italia e nel mondo.
Gli speciali dell'anno
Gli "speciali" sono l'elemento portante, in **Informatica Oggi**, più che in qualsiasi altra rivista Jackson.
Redatti negli Stati Uniti, su misura per il lettore italiano. Una garanzia di serietà e professionalità.
Argomenti dei servizi speciali pubblicati su **Informatica Oggi** nel 1982: Modem e Multiplexer - Software Tool - Ethernet, una realtà di mercato? - PBX e Reti Locali: la lotta per l'ufficio integrato - Mini & Office Automation - NCC: dai dinosauri alle formiche - Mainframe IBM e compatibili - Data Base Machine - Il boom dei sistemi fault-tolerant - Small Talk e Interlisp: una nuova interfaccia uomo-macchina.
INFORMATICA OGGI
LA RIVISTA DI ELABORAZIONE DATI E TELEMATICA
ANNO 2 N. 9-'82 L. 3.000
LA FAMIGLIA 308X
L'IBM 3082, elaboratore di controllo comune a tutte le cinque unità della famiglia 308X.

Bit
La prima rivista europea di personal computer, software e accessori.
Anche con Bit, il Gruppo Editoriale Jackson è stato lungimirante.
È curioso (e ci rende orgogliosi) il fatto che Bit sia la prima rivista europea di personal, home e business computer.
Una pubblicazione stimolante per chi vuol vivere l'affascinante avventura tecnica offerta dal mondo dei piccoli sistemi.
In un mercato in crescente espansione, Bit è divenuta ormai il leader incontrastato sul piano editoriale: per la serietà, completezza e tempestività dei suoi "Test", per i programmi dedicati pubblicati ogni mese, per il ruolo formativo che la rivista ha assunto fin dall'inizio, accompagnando nella crescita culturale e tecnica l'hobbista, il tecnico, l'appassionato di personal computer.
È la più letta tra le riviste Jackson.
Una miniera di idee e soluzioni pratiche per "giocare" con il computer, imparare a programmare, disporre di un'informazione approfondita, su tutte le ultime novità del mercato.
Gli speciali dell'anno
Gli speciali di Bit sono i Test sulle ultime novità nei personal, o meglio i BITEST, divenuti ormai un classico, per il metodo di analisi, sia hardware sia software, le fotografie, il contenuto tecnico e professionale dell'informazione, i raffronti.
In più, quattro volte all'anno, un numero di Bit è monografico. Questi i temi dell'ultimo anno: Computergrafica - Word Processing - Pocket Computer - Bit Didattica.
Bit
DIDATTICA CON IL PERSONAL COMPUTER
DALL'AMERICA MUSIC SYSTEM
MINI ROBOT: IL PRIMO PERS
ROBOT
Bitest:
Digital Rainbow 100
Nella foto il sistema Digital, unità centrale, stampante video e tastiera. Sulla sinistra l'interessante opzione "Thelephone nagement System" disponibile sul Professional.

PERSONAL SOFTWARE

La prima rivista europea di software per personal computer.

Personal Software apre un capitolo nuovo nell'editoria legata ai personal computing: è una rivista di software, dedicata a due tipi di lettori: coloro che già posseggono un personal (o intendono acquistarlo entro breve) e coloro che, pur non possedendo un personal, si interessano di software in BASIC, di programmi e sistemi operativi per personal.
Di personal, in Italia, ne esistono circa 30000: un'ottima base di lettori da cui, fin dal primo numero, uscito nel giugno '82, sono provenuti una miriade di suggerimenti e consigli.
Personal Software è una rivista attualissima, interattiva con il lettore, realizzata con una formula del tutto nuova.
Una prima sezione è dedicata ad articoli generali, di approfondimento teorico di certi aspetti e problemi di software. La seconda sezione è fatta di programmi, già predisposti, testati e pubblicati insieme ai relativi listati, dedicati ai personal più diffusi sul mercato.
Quando è tecnicamente possibile, lo stesso programma è pubblicato nelle varie versioni, relative alle macchine su cui può operare.
Personal Software è la rivista per i veri "amatori" dei personal, per un pubblico giovane, intelligente, dinamico come il mezzo stesso a cui si rivolge.

Gli speciali dell'anno

Personal Software è appena nata: non si può quindi pubblicare un elenco di speciali. Ricordiamo, tuttavia, alcune iniziative che già hanno riscosso notevole successo: la pubblicazione, in omaggio, della GUIDA AL SOFTWARE pubblicato sulle riviste italiane; una testimonianza del servizio informativo completo che il Gruppo Editoriale Jackson intende proporre, con questa nuova iniziativa.

22 numeri
L. 35.000
anziché
L. 44.000
11 numeri
L. 31.000
anziché
L. 38.500
8 numeri
L. 19.000
anziché
L. 24.000
12 numeri
L. 24.500
anziché
L. 30.000
GRUPPO
EDITORIALE JACKSON
SERVIZIO ABBONAMENTI
l'ELETTRONICA
ELETTRONICA OGGI
AUTOMAZIONE OGGI
ELEKTOR
Alcuni esempi
EO + l'E L. 64.000
EO + AO L. 48.000
EO + IO L. 55.500
IO + BT L. 50.500
IO + l'E L. 59.500
CW + IO L. 84.500
BT + PS L. 52.000
CW + l'E L. 93.000
EO + l'E + EK L. 86.500
EO + l'E + IO L. 88.500
EO + l'E + BT L. 88.000
IO + BT + PS L. 76.500
BT + IO + l'E L. 83.500
EO + l'E + EK + AO L. 102.500
tutte le riviste ... L. 249.000
LEGENDA
l'E = l'ELETTRONICA
EO = ELETTRONICA OGGI
AO = AUTOMAZIONE OGGI
EK = ELEKTOR
IO = INFORMATICA OGGI
CW = COMPUTERWORLD
BT = BIT
PS = PERSONAL SOFTWARE
SM = STRUMENTI MUSICALI
CAMP
ABBONA

ABBONAMENTO CUMULATIVO A DUE O PIU' RIVISTE CON SCONTO PARTICOLARE

Tutti coloro che sottoscrivono abbonamenti a due o più riviste godono di un prezzo ulteriormente agevolato, come appare nella seguente tabellina.

Abbonamento a due riviste somma dei prezzi scontati delle due riviste - L. 2.000.
Abbonamento a tre riviste somma dei prezzi scontati delle tre riviste - L. 4.000.
Abbonamento a quattro riviste somma dei prezzi scontati delle quattro riviste - L. 7.000.
Abbonamento a cinque riviste somma dei prezzi scontati delle cinque riviste - L. 10.000.
Abbonamento a sei riviste somma dei prezzi scontati delle sei riviste - L. 13.000.
Abbonamento a sette riviste somma dei prezzi scontati delle sette riviste - L. 16.000.
Abbonamento a otto riviste somma dei prezzi scontati delle otto riviste - L. 20.000.
Abbonamento a nove riviste somma dei prezzi scontati delle nove riviste - L. 25.000.

N.B. - **Per sottoscrivere abbonamenti utilizzate il modulo di c.c.p. inserito in questo fascicolo oppure inviate un assegno o un vaglia postale al nostro ufficio abbonamenti.**

AGNA
MENTI

IL TASTO DEL RISPARMIO.

GRANDE CONCORSO J

IL SUPERPREMIO PER TUTTI ...

Un meraviglioso viaggio nella Silicon Valley

A sud di questa baia c'è la favolosa Silicon Valley: il paradiso della microelettronica e dell'informatica. Quasi tutte le industrie "che contano" ci sono: anche il Gruppo Editoriale Jackson, con la propria sede di Sunnyvale. Tra tutti gli abbonati sarà sorteggiato un viaggio soggiorno della durata di una settimana.

Sarete ospiti della GEJ Publiching Group, visiterete la splendida e soleggiata California.

L'ELETTRONICA
Apple II - Uno dei più diffusi e prestigiosi personal computer. Infinite possibilità di utilizzo. 48 Kbyte RAM

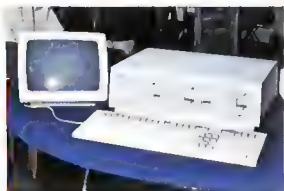
ELETTRONICA OGGI
TEK 2213 - L'oscilloscopio Tektronix a 2 canali DC 60 MHz - 20 mV/div. 50 MHz 2 mV/div. Il sogno di ogni tecnico e laboratorio elettronico.

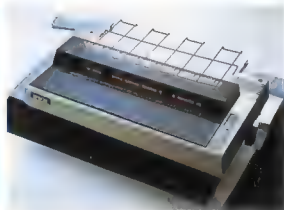
REGOLAMENTO DEL CONCORSO

1) Il Gruppo Editoriale Jackson srl promuove un concorso a premi in occasione della campagna abbonamenti 1983
2) Per partecipare al concorso è sufficiente sottoscrivere un abbonamento 1983 ad almeno una delle nove riviste Jackson entro il 28 2 1983
3) È previsto un premio (viaggio soggiorno) da sorteggiare fra tutti gli abbonati a nove premi uno per ciascuna rivista, da sorteggiare fra gli abbonati alle singole riviste
4) Gli abbonati a più di una rivista Jackson avranno diritto all'inserimento del proprio nominativo per l'estrazione relativa al viaggio soggiorno tante volte quante sono le riviste cui sono abbonati
5) L'estrazione dei premi indicati in questo annuncio avverrà presso la sede Jackson entro il 30 6 1983
6) L'elenco dei vincitori e dei premi sarà pubblicato su almeno sei delle riviste Jackson subito dopo l'estrazione. Il Gruppo Editoriale Jackson inoltre, ne darà comunicazione scritta ai singoli vincitori
7) I premi verranno messi a disposizione degli aventi diritto entro 60 giorni dalla data di estrazione
8) I dipendenti, i familiari, i collaboratori del Gruppo Editoriale Jackson sono esclusi dal concorso

GRUPPO EDITORIALE JACKSON
SERVIZIO ABBONAMENTI

... E PER OGNI RIVISTA

AUTOMAZIONE OGGI
Mini Robot - Il Robot in kit della Soft-Power. Una periferica per personal computer dalle infinite applicazioni per esplorare il nuovo mondo della robotica. A portata di "Basic".

COMPUTER WORLD
Rainbow 100 - Il superbo computer Digital Equipment al vertice della gamma personal. Doppio processore, da 64 a 256 Kbyte RAM, 2 floppy disk da 600 Kbyte.

ELEKTOR
Junior Computer - Il computer didattico in kit che ha entusiasmato gli hobbisti di tutti i paesi europei.

INFORMATICA OGGI
Epson MX100 - La stampante a impatto famosa in tutto il mondo. Massima affidabilità e ottime prestazioni. Una periferica d'eccezione.

BIT
Spectrum - Il nuovo entusiasmante personal Sinclair. Incredibili capacità grafiche a colori. Un gioiello di tecnologia e miniaturizzazione.

PERSONAL SOFTWARE
VIC 20 - Un best-seller nei personal. Il sistema ideale per divertirsi in modo intelligente con il computer.

STRUMENTI MUSICALI
Roland HP 70 - Il pianoforte elettronico portatile con prestazioni professionali. 75 tasti, effetto chorus, touch-control per la dinamica su ogni tasto

RISERVATO A CHI SI ABBONA ENTRO IL 28-2-'83

AUT. MIN. N° D.M. 4/236724 del 27/10/82

ENCICLOPEDIA DI E

un'opera unica, completa, rigorosa, aggiornata, ma fa

Un'opera seria perché l'Elettronica e l'Informatica sono una cosa seria

L'Enciclopedia di Elettronica e Informatica, composta da 50 fascicoli pubblicati settimanalmente, sarà disponibile a partire da gennaio 1983 in tutte le edicole a L. 2.500 al fascicolo.

Ogni fascicolo è costituito da:

- 12 pagine di Elettronica Digitale - Microprocessori;
- 16 pagine di Elettronica allo stato solido - Telecomunicazioni oppure 16 pagine di Informatica - Informatica e Società;
- 1 scheda di Elettrotecnica.

I fascicoli saranno raccolti in 7 volumi di 200 pagine l'uno più 1 raccoglitore per le 50 schede di Elettrotecnica.
Copertine con sovracoperte, risguardi e indici L. 5.000.
Raccoglitore per le 50 schede L. 5.000.

1500 pagine, 700 foto a colori, 2200 illustrazioni a colori

ELETTRONICA E INFORMATICA

ELETTRONICA DIGITALE

ELETTRONICA DIGITALE

INFORMATICA

INFORMATICA

LETTRONICA E INFORMATICA

acile e scorrevole, che tutti possono capire

Realizzata in collaborazione con il Learning Center Texas Instruments

Se desiderate abbonarvi all'Enciclopedia di Elettronica e Informatica usufruendo di un prezzo speciale, ricevendo direttamente a casa vostra le copie, potete inviare un assegno, o un vaglia postale oppure versate l'importo di L. 130.000 (anziché 165.000) sul ccp nº 11666203 intestato a Gruppo Editoriale Jackson - Grandi Opere.

Per evitare danni ai fascicoli e garantire il recapito, le spedizioni saranno effettuate mensilmente (raggruppando 4 o 5 fascicoli) in apposito imballo.

ELETTROTECNICA
• Costituzione della materia • Conduzione, resistività, ecc. • Corrente-Tensione-Resistenza • Circuito elettrico • Kirchhoff ed altri metodi risolutivi (Thevenin, Norton, Sovrapposizione) • Lavoro, Potenza, Rendimento • Campo magnetico • Campo elettrico • Circuito Magnetico • Induzione e Autoinduzione • Bobina • Condensatore • Corrente Trifase • Potenza Trifase • Amperometri, Voltmetri, altri strumenti di misura • Funzionamento del trasformatore • Generatore, motore • Motore a c.c.

ELETTRONICA ALLO STATO SOLIDO
• Principi fisici dei tubi • Triodo • Diodo a semiconduttori • Curve caratteristiche diodo e impieghi • Transistori • SSI, LSI, VLSI, Gate Array • Tecnologie elettroniche (Bipolari, Mos, Cmos) • FET, MOSFET • SCR, DIAC, TRIAC • Optoelettronica (LED, LCD, CCD, Plasmadisplay,. .) • Relais • Protezioni • Fotocellule, Fotodiodi, Termistori, Pannelli solari • Touchcontrol

ELETTRONICA DIGITALE Vol. 1
• AND or NOT • Sistemi di numerazione • Codici • Algebra di Boole • Karnaugh • Codificatori • Decodificatori • Matrici • Selettori • Multiplexer • Comparatori • Addizionatori • Sottrattori • RTL, DTL, TTL, FST, TTL S • NMOS, PMOS, VMOS, CMOS, I²L • flip-flop, SR, JK, T, D • Multivibratori (Schmitt) • Registri • Dispositivi per sintesi vocale e per Speech Recognition

ELETTRONICA DIGITALE VOL. 2
• Shift register • Clock • Contatori Binari • Contatori Decimali • Tipi di memorie • ROM, RAM, EPROM • Organizzazione della memoria • Operazionali • Sample and hold • Convertitori A/D e D/A • Conversione V-f/f-V • Calcolo delle probabilità • Struttura del Bus • Bus standard • Trasmissione dati • Interfacce standard • Optocoupler • Fibre ottiche • Esempi (UART - USART UIA)

MICROPROCESSORI
• I/O di un microprocessore • Struttura di un microprocessore • Interfacce specializzate • Mezzi di sviluppo per microprocessori • Linguaggi • Indirizzamento • Programmazione • Microcalcolatori • Microprogrammazione • Sviluppi Futuri

TELECOMUNICAZIONI
• Onde elettromagnetiche • Filtri - antenne - radar • Trasmissione: *modulazione, trasduttori cavi, acustica, ottica, trasmissione dati (cenni), comandi a distanza, controllo di parità* • Ricezione: *Radio, TV, Telefonia, CB* • Trasmissione dati • Varie

INFORMATICA DI BASE
• Informatica: ieri, oggi e domani • Architettura del calcolatore elettronico digitale • Funzionamento del calcolatore • Le memorie • Tecniche e dispositivi di ingresso/uscita • Struttura dei dati • Gli archivi dei dati • Programmazione • Sistemi operativi • Linguaggi e traduttore • Assembler • Cobol • Basic • Fortran • Pascal • Simula • Lisp • PL1 • RPG • Altri linguaggi • I data base

INFORMATICA E SOCIETA'
• Il computer e la scienza • Il computer e la tecnica • Il computer e la vita di tutti i giorni • Il computer e l'elettronica nell'abitazione • Il computer e l'ufficio • Il computer e l'elettronica nella produzione • Il computer e l'elaborazione nella musica • La computergrafica • La progettazione e il controllo tramite il computer • I problemi di segretezza e di esclusività • Computer ed intelligenza artificiale • Computer e istruzione

LIBRI JACKS
LA MIGLIORE FORMAZ
NELL'ELETTRONICA E NEL

I prezzi blu sono riservati agli abbonati

ON.
ZIONE TECNICA
L'INFORMATICA.

GRUPPO EDITORIALE JACKSON

Divisione Libri

CORSO PROGRAMMATO DI ELETTRONICA ED ELETTROTECNICA

40 volumi 2700 pagine

Cod. 099A
L. 109.000 anzichè L. 129.000
Abbonati L. 87.200

Il corso articolato in 40 volumi per complessive 2700 pagine, permette in modo rapido e conciso l'apprendimento dei concetti fondamentali di elettrotecnica ed elettronica di base, dalla teoria atomica all'elaborazione dei segnali digitali.
La grande originalità dell'opera non risiede solo nella semplicità con cui gli argomenti vengono trattati, anche i più difficili, non solo nella struttura delle oltre 1000 lezioni incentrate su continue domande e risposte, esercizi, test, al fine di permettere la costante valutazione del grado di apprendimento raggiunto, ma soprattutto nella possibilità di crearsi in modo organico un corso "ad personam" rispondente alle singole necessità ed obiettivi.
Se non avete tempo o non volete dedicare 120 delle vostre ore, anche in modo frammentario, al completamento del corso, potete seguire un programma di minima, sempre con brillanti risultati, con obiettivi, anche parziali, modificabili dinamicamente nel corso delle letture successive.
Ogni libro è una monografia esauriente singolarmente consultabile per l'approfondimento di un particolare argomento.

* ATTENZIONE. Per ordinare questi libri utilizzare l'apposita cedola di commissione libraria. L'OFFERTA È VALIDA SOLO FINO A 28/2/1983. Dopo tale data gli abbonati avranno comunque diritto allo sconto del 10% su tutti i libri JACKSON, novità comprese. I libri elencati possono essere ordinati anche dai non abbonati utilizzando la stessa cedola di commissione libraria. In questo caso naturalmente non si avrà diritto a sconto alcuno.

INTERFACCIAMENTO DI MICROCOMPUTER

Manuale pratico di REGISTRAZIONE MULTIPISTA

MANUALE PRATICO DEL RIPARATORE RADIO-TV

Cod 701P
L 18.500
L 14.800

CEDOLA DI COMMISSIONE LIBRARIA

Da inviare a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap.   Città   Provincia

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato nella vostra offerta speciale + L. 1.500 per contributo fisso spese di spedizione
☐ Allego assegno n° .... di L. .... (In questo caso la spedizione è gratuita)

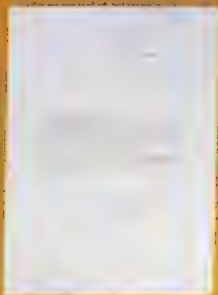
☐ Non abbonato ☐ Abbonato sconto 20% ☐ l'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor
☐ Informatica Oggi ☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali

Data ______________________ Firma ______________________

TechnoClub

PRESENTA

applemail*

Finalmente risolto il problema delle circolari e delle lettere personalizzate.

Con Apple Writer puoi comporre il testo della circolare

Con il Personal Data Base generare archivi o utilizzare quelli che già usi per altri scopi

L'estrema semplicità di utilizzo rendono questo package lo strumento ideale per redigere circolari promozionali, lettere personalizzate, estratti conto periodici o qualunque altro documento contenente dati e indirizzi.
Tutto ciò senza dover creare archivi complementari, ma sfruttando quelli già creati e utilizzati normalmente per altri servizi. Il programma, fornito completo di manuale d'uso, floppy disk ed elegante contenitore, può essere acquistato al prezzo di L. 150.000 (IVA compresa) presso uno dei rivenditori Apple oppure presso TechnoClub inviando il coupon riprodotto in questa pagina
In entrambi i casi diventerai automaticamente socio del TechnoClub con tutti i vantaggi illustrati nelle pagine precedenti
Possiamo inoltre fornire i programmi
Apple Writer a L.125.000
Apple Personal Data Base a L. 69.000

Con estrema semplicità AppleMail consente di creare circolari o lettere personalizzate

Cognome ....
Nome
Via .... N ....
Città .... CAP ....
C F (per le aziende)
Confermo l'acquisto di

package AppleMail a L. 150.000 cad.
package Apple Writer a L 125.000 cad.
package Personal Data Base a L 69.000 cad.

DATA .... FIRMA ....

scegliendo la seguente forma di pagamento:

- ☐ Allego assegno N .... di L ....
- ☐ Ho versato l'importo tramite vaglia postale di cui allego ricevuta
- ☐ Ho versato l'importo sul CCP n 19445204 intestato a TechnoClub - Milano

NON SI EFFETTUANO SPEDIZIONI IN CONTRASSEGNO

*Apple è un marchio registrato della Apple Computer Inc Cupertino - USA

TechnoClub - Via Rosellini, 12
20124 Milano - Tel. (02) 6888228

Raccolta di routine Basic

MODIFICHE E COMMENTI PER LA ROUTINE 1

Sono giunte un gran numero di lettere relative al primo numero di questa rubrica in cui si parlava della punteggiatura. Sono state proposte modifiche, miglioramenti, e sono state messe in rilievo le diversità di comportamento dei vari computer.

Anzitutto, una precisazione di carattere generale: la routine di punteggiatura (come tutte quelle che seguiranno) è scritta in Microsoft Basic, di cui, come si sa, esistono diverse versioni. Noi utilizziamo quella standard, descritta per esempio in *Microsoft Basic* (Franco Muzzio editore). Evitiamo inoltre di usare funzioni non disponibili sui personal più diffusi (come ELSE). Ne segue che le routine che pubblichiamo sono abbastanza generali per poter essere eseguite sulla maggior parte dei personal computer in commercio. Naturalmente, l'implementazione su di un particolare computer potrà essere fatta più brevemente o sfruttando particolari caratteristiche dello stesso. Qualche computer potrà poi necessitare di alcune modifiche. Su tutto questo invitiamo i lettori a scriverci, sottoponendoci le versioni per il loro modello. Questa volta, intanto, abbiamo già ricevuto delle segnalazioni relative a PET, TRS-80 e Apple.

PET e VIC

La routine di punteggiatura funziona su tutti i PET e VIC per numeri con non più di nove cifre. Il signor Edilio Fazzi di Cengio, Savona, suggerisce alcune modifiche per permettere la punteggiatura di numeri con più di nove cifre. Ecco la sua routine.

```
1 INPUT N$,X
2 GOSUB 1000
3 GOTO 1
1000 :
1010 REM  SCOPO:
1020 REM  METTE LA PUNTEGGIATURA ALL'ITALIANA NEI NUMERI
1030 REM  INTERI CON PIU' DI 3 CIFRE E LI INCOLONNA A
1040 REM  DESTRA IN UNA POSIZIONE DETERMINATA.
1050 :
1060 REM  VARIABILI:
1070 REM  N$  NUMERO DA PUNTEGGIARE (INPUT)
1080 REM  X   POSIZIONE DI INCOLONNAMENTO (INPUT)
1090 REM  N   VALORE NUMERICO DEL NUMERO PUNTEGGIATO (OUTPUT)
1100 REM  X$  NUMERO PUNTEGGIATO (OUTPUT)
1110 :
1120 REM  VINCOLI:
1130 REM  VALIDA PER X>INT(C/3)+C DOVE C E' IL NUMERO [illegible]
1140 REM  (VINCOLO NON CONTROLLATO).
1150 :
1160 REM  **********   INIZIO ROUTINE   **********
1170 :
1180 N=VAL(N$)
1190 N$="  "+N$
1200 X$=""
1210 L=LEN(N$)-2
1220 IF L>2 THEN 1250
1230 I=L
1240 GOTO 1280
1250 FOR I=L TO 3 STEP -3
1260 X$="."+MID$(N$,I,3)+X$
1270 NEXT
1280 X$=LEFT$(N$,I+2)+X$
1290 PRINT TAB(X-LEN(X$))X$
1300 RETURN
```

La routine funziona solo per numeri che vengono introdotti come stringa: cosa succede se il numero da punteggiare è il risultato di un calcolo, e quindi è disponibile solo in formato esponenziale?

TRS-80 mod. 1

La routine di punteggiatura funziona per i numeri con non più di sei cifre. Il signor Rosario Bizioli di San Polo, Brescia, propone di aggiungere un segno dichiarativo # di doppia precisione alla variabile N nelle righe 1 e 1090, e inoltre di aggiungere 0 CLEAR 100. Ciò permette di punteggiare numeri fino a 16 cifre.

Apple II

La routine di punteggiatura funziona per i numeri con non più di nove cifre, ma ai numeri 1000 e 1000000 non viene aggiunto il punto dopo la prima cifra. Ciò dipende dal fatto che, contrariamente a PET e TRS-80, l'Apple non prevede l'aggiunta di uno spazio a sinistra nella trasformazione da numero a stringa. Per i calcolatori che funzionano come l'Apple, il signor Alcide Andreatta di Castelcucco, Treviso, propone di eseguire queste modifiche:

```
1120 IF L > = 2 GOTO 1150
1150 FOR I=L TO 2 STEP - 3
```

Preghiamo coloro che ci scrivono di precisare in apertura di lettera a quale tipo e modello di computer si riferiscono e, se propongono modifiche ai programmi pubblicati, specificare quali sono i vantaggi che tali modifiche consentono, o gli errori che tali modifiche correggono.

Invitiamo inoltre tutti coloro che hanno modelli diversi di computer a sottoporci la versione della routine adatta alle loro macchine (in particolare Atari, ZX80 e 81, Atom, ecc.)

Indirizzate a:
PERSONAL SOFTWARE
Via Rosellini, 12
20124 Milano

Generatore di labirinti

di *C. Bond*

Qui è descritto un algoritmo soprendentemente corto che produce labirinti casuali di ogni formato direttamente sul video. Il programma può funzionare su ogni microcomputer che permetta la grafica indirizzata in memoria. Sono forniti dettagli per l'applicazione su PET/CBM. Un tipico labirinto generato da questo programma è riportato in figura 1.

Per comprenderne il funzionamento, ci si riferisca al diagramma di flusso di figura 1 e alla lista del programma. Le spiegazioni seguenti dovrebbero chiarire i dettagli.

Il campo di base

L'algoritmo opera su un campo di base che deve essere generato sullo schermo prima della linea 200 nel programma 1. Il campo consiste di un numero dispari di righe orizzontali, ognuna con un numero dispari di celle: una matrice rettangolare. È conveniente pensare al campo come a una matrice bidimensionale con l'angolo superiore sinistro avente coordinate X=0 e Y=0, dove X è la direzione orizzontale e Y quella verticale. Il programma non usa le coordinate per identificare locazioni assolute, ma il concetto è utile nel configurare il campo.

Dato che la cella in alto a sinistra del campo ha coordinate 0,0 le coordinate terminali sia orizzontali che verticali dovranno essere numeri pari. Inoltre, il campo di base deve essere circondato da tutte le parti da celle di memoria i cui contenuti sono differenti dai numeri usati per identificare il campo. Cioè, se il campo è costituito da spazi inversi (*reverse video*), i numeri corrispondenti a questo carattere non devono essere visualmente adiacenti al campo.

Ciò può succedere inavvertitamente se la RAM di schermo e la ROM di sistema hanno indirizzi contigui. Una precauzione sufficiente consiste nell'evitare di coprire l'intero schermo con il campo. Si lasci almeno uno spazio all'inizio e alla fine di ogni linea, e non si occupino la prima e l'ultima linea.

Il generatore di labirinti

La creazione del labirinto inizia ponendo uno speciale contrassegno in una opportuna casella d'inizio. Il programma qui presentato comincia sempre dalla casella corrispondente alle coordinate 1,1. Ogni cella con coordinate dispari va bene, purché sia all'interno del campo.

Quindi, si sceglie una direzione a caso, generando un numero intero casuale tra 0 e 3. Questo intero, mediante una piccola tavola, determina una direzione ed una corrispondente cella distante esattamente due celle da quella attuale. Questa nuova cella viene esaminata (mediante PEEK) per vedere se fa parte del campo. Se sì, l'intero viene messo in questa cella come contrassegno, e la barriera tra esso e la cella attuale viene eliminata.

Inoltre, il puntatore alla cella attuale viene ora spostato in modo che punti alla nuova. Questo processo viene ripetuto fino a che la nuova cella non supera il test, cioè non è una cella di campo. Quando ciò succede, il vettore di direzione viene ruotato di 90 gradi e il test viene ripetuto. Quindi, il cammino ricavato nel campo continua fino a che si raggiunge un "vicolo cieco". Quando vi si capita, possiamo far uso dei contrassegni che sono stati tolti lungo la strada.

Questi possono essere verificati per determinare da quale direzione veniamo, in modo da poter ritornare e ricercare nuovi cammini. Fino a che non se ne trovano, il programma torna indietro, un passo alla volta, cancellando il contrassegno che trova. Quando si può prendere una nuova direzione, il puntatore viene posizionato in quella direzione, e il processo continua come sopra.

Alla fine, il puntatore ritorna all'inizio, una condizione che viene individuata dal ritrovamento dello speciale contrassegno d'inizio (ora di fine).

Il programma

La lista 1 contiene il programma completo implementato sul PET, ma applicabile anche ad altre macchine. La tavola delle direzioni predisposta nelle linee 100 e 110 converte un intero in un offset d'indirizzo. In questo caso (schermo a 40 colonne) ci interessa fare passi di due celle a destra, in alto, a sinistra o in basso. Gli indirizzi di memoria di queste celle differiscono da quello della cella attuale per 2, −80, −2 e 80 rispettivamente. Per computer con display di 64 colonne, l'80 deve essere sostituito da

128.
La linea 120 contiene le variabili dipendenti dalle macchine. SC è l'indirizzo di memoria dell'inizio dello schermo.

Le linee 130-160 stabiliscono il campo di base sullo schermo. Per il PET scegliamo 23 righe di 39 colonne ognuna. Il resto del programma traccia il labirinto, nel modo discusso precedentemente. La linea 310 è semplicemente un punto di stop che evita lo spostamento dello schermo.

Può non essere immediatamente ovvio che questo algoritmo produca un labirinto con un unico cammino non banale tra due punti, o che il labirinto venga sempre riempito completamente, ma lo si può dimostrare. Sebbene non sia questo il luogo adatto per la dimostrazione, può essere interessante notare che per un labirinto di qualunque formato ci sono esattamente

$$\frac{(H-1)(V-1)}{2} - 1$$

celle vuote nel labirinto completato, dove H è il numero di celle in ogni riga del campo e V il numero delle righe.

Una interessante caratteristica di questo algoritmo è che funziona ugualmente bene in certi tipi di campi non rettangolari. Campi ad U o campi con buchi sono possibili, sempre che siano rispettate alcune regole. Basta assicurarsi che le coordinate delle celle in alto a sinistra e in basso a destra di ogni "foro" siano coppie di numeri dispari. Inoltre, se c'è un'unica riga di celle tra un "foro" e l'esterno del campo, può essere rimossa. Vedi la figura 2.

```
10 REM ******************************************
20 REM *                                        *
30 REM *        GENERATORE DI LABIRINTI         *
40 REM *        =======================         *
50 REM *                                        *
60 REM *          VERSIONE [illegible]          *
70 REM *                                        *
80 REM ******************************************
100 DIM A(3): REM PREPARA LA TABELLA DIREZIONI
110 A(0)=2: A(1)=-80: A(2)=-2: A(3)=80: REM VALORI
    PER SCHERMO DI 40 COLONNE
120 WL=160: HL=32: SC=32768: A=SC+81: REM VALORI
    PER PET COMMODORE
130 PRINT CHR$(147): REM CANCELLA LO SCHERMO E
    PREPARA LO SFONDO DEL LABIRINTO
140 FOR I=1 TO 23
150 PRINT CHR$(18);
153 [illegible]
155 [illegible]
157 NEXT W: [illegible]
160 NEXT I
200 REM GENERA IL LABIRINTO
210 POKE A,4
220 J=INT(RND(1)*4): X=J
230 B=A+A(J): IF PEEK(B)=WL THEN POKE B,J: POKE
    A+A(J)/2,HL: A=B: GOTO 220
240 J=(J+1)*-(J 3): IF J  X THEN 230
250 J=PEEK(A): POKE A,HL: IF J<4 THEN A=A-A(J):
    GOTO 220
300 REM LABIRINTO PRONTO! ASPETTA CHE VENGA
    PREMUTO UN TASTO
310 GET C$: IF C$="" THEN 310
```

Programma 1.

```
1000 REM [illegible]
1010 POKE A,81: J=2
1020 B=A+A(J)/2: IF PEEK(B)=HL THEN POKE
     B,81: POKE A,HL: A=B: J=(J+2)+4*(J 1)
1030 J=(J-1)-4*(J=0): GOTO 1020
```

Programma 2.

Il topo

Con piccole modifiche il generatore di labirinti può diventare un topo artificiale. Il programma 2 descrive una routine che può esere aggiunta al generatore di labirinti e che crea un topo che attraversa il labirinto senza fine. Il topo obbedisce alla "regola della mano sinistra" quando è possibile la scelta di una direzione. In altre parole, quando si trova alle prese con un punto di diramazione, esso, se possibile, gira a sinistra. Altrimenti, va avanti. Se non è possibile alcuna scelta, torna indietro.

Fig. 1.

Fig. 2.

PICCOLI ANNUNCI

Scambio **programmi** di tutti i generi, ma specialmente giochi per il nuovo personal computer della Texas Instruments: il TI 99/4A
Stefano Oggioni - Via Verdi, 31 - Bussero (MI) - Tel 9500061 diverrà 95039061)

Scambio **programmi** gestionali, magazzino per PET/3032, ingegneria civile ecc. con programmi completi per paghe e stipendi con gestione contratti di lavoro diversi. Telefonare ore 19-20
Paolo Stella - Via Strada 62, 28 - 67100 L'Aquila - Tel 23273

Cerco **utenti** di VIC-20 per scambio informazioni, materiale, programmi, etc. Assicuro massima serietà.
Ernesto Bastoni - Via Rocca D'Anfo, 2 - 25100 Brescia - Tel 030/305906

Vendo **ZX81-ZX80** nuova ROM bellissimi programmi veramente originali molto elaborati (i migliori dei miei due anni di hobby) vendo ad un prezzo ridicolo: un nastro BASF/TDK 60 minuti colmo di programmi per sole L. 12 000 compreso costo nastro e postali! Meravigliosi programmi elaborati da me per cui garantisco soddisfazione, a meno di 500 lire l'uno.
Bruno Del Medico - Via Torino, 72 - 04016 Sabaudia (LT)

Vendo **test** psicodiagnostici solo a medici o psicologi che comprovino titolo di laurea. Sono scritti in MBASIC (anche per CP/M) e forniti in dischi 5"
Vincenzo Colaciuri - Via Monte Bianco, 10/12 - 24020 Scanzorosciate (BG) - Tel 659049

Cerco **persone** disposte a scambiare programmi e idee sul Sinclair ZX-81. Telefonare o scrivere a
Paolo Ghedini - Via Brioschi, 33 - 20136 Milano - Tel. 02/8372839

Vendo **programmi** per ZX computers: assolute novità, registrati su cassetta: favoloso scacchi II 7 livelli difficoltà - velocissimo - scacchiera grande L. 15.000 - Mazogs eccezionale grafica e movimento L. 12.000 - Scramble il nuovo gioco che pilota la sound board L. 9.000 - Galaxian - asteroid e tantissimi altri. Elenco a richiesta
Massimo Soncini - Via Monte Suello, 3 - 20133 Milano - Tel 02/727665

Se possiedi un PET e vuoi scambiare **programmi** inviami una tua cassetta contenente i tuoi ed io la contraccambierò con i miei sia gestionali che giochi.
Luigi Menghi - Via Maranello, 7 - 00125 Acilia (Roma) - Tel. 06/6069235

Scambio **soft** ZX81, zona Modena.
Manfredini - C.P. 25 - Modena

Cerco possessori di VIC 20 che abbiano sviluppato o abbiano intenzione di sviluppare **programmi** in L.M. GS02 o che abbiano "esplorato" la pagina della memoria
Peppe D'Ottavo - Via Donati, 10 - Milano - Tel. 02/425493

Vendo **cassetta** programmi gioco per VIC 20 e altri programmi chiedere lista a
Claudio Marchiondelli - Via Libertà 3 - 33010 Raspano (UD)

Scambio **software** per VIC 20 - vendo ZX80 + 16 Kbyte RAM + alim + cavi + garanzia da convalidare + istr. bilingue - assemblato in fabbrica - Vinto ad un concorso L. 300 000 usato 10 giorni
Amos Aimi - Via Zanella, 11 - 43015 Noceto (PR)

Cambio/vendo **programmi** per VIC 20 vendo cassetta con 5 giochi per VIC 20 L. 15 000. Per VIC 20 vendo cass. con equo canone + isolamento termico L. 15 000
Maurizio Mellone - Via Sabbionara 9 - 36061 Bassano del Grappa (VI) - Tel 20015

Atari **Club** per scambio programmi, più di cento programmi animazione grafica, musica, games, uno più favoloso dell'altro scrivetemi per saperne di più. W gli ataristi!!
Marco Rosati - Via Giovanni XXIII - 10040 Caprie (TO)

Cambio **programmi** per Apple II e TRS/80, gestionali, utilities, giochi vari. Inviare lista invierò la mia
Giovanni Carrella - Via Feline, 11/C - 84100 Salerno - Tel 089/354854

Cambio/vendo **software** per Apple II
Marco Montresor - P.zza C. Alberto, 25 - 37067 Valeggio S/M (Verona)

PICCOLI ANNUNCI PERSONAL SOFTWARE

Sei un lettore di PERSONAL-SOFTWARE e vuoi entrare in contatto con tutti gli altri lettori per comperare, cambiare o vendere software? Spediscì questo tagliando a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome. .. Cognome

Via CAP

Città Tel

Dove trovare il 99/4A della Texas Instruments

SANREMO UNELCO 0184/883832 • **SAVONA** SPERATI 019/25875 • **IMPERIA** TELEROS 0183/24314 • **GENOVA** CLU 010/510355, COMPUTER CENTER 010/300797, BENNATI 010/298476, COMPUFFICIO 010/564493, FOTOMONDIAL 010/280322 LA NASCENTE 010/291641 SALVIATI 010/595750. • **LAVAGNA (SP)** COMPUTER CENTER 0185/314142 • **LA SPEZIA** BARBAGALLO 0187/35275 • **NOVI LIGURE (GE)** ELCAMA 0143/743687 • **CUNEO** CASATI 0171/3114 • **BRA (CN)** MAGAZZINI BONA 0172/412778 • **ALESSANDRIA** VERONELLI 0131/53462 • **TORINO** MISTER ELECTRONICS 011/510353, ABA 011/389328, CSC 011/532730, CELID 011/540875, ELEDRA 3S 011/3099111, MECCANOGRAFICA TORINESE 011/238803 LA MERIDIANA 011/7497752 • **BIELLA (VC)** TEOREMA 015/777179 • **NOVARA** IMES 0321/24208 • **AOSTA** LAGOVAL 0165/43929 • **VARESE** BERNASCONI 0332/29186 • **GRAVINA (VA)** SCF 0332/240250 • **GALLARATE (VA)** PROFIT CENTER 0331/799019 SED 0331/795735 • **BERGAMO** FRABERT ELETTRONICA 035/248362 • **ALBINO (BG)** IL DATO 035/752225 • **BRESCIA** VILLA E VENTURINI 030/345591 VIGASIO 030/53588 • **MILANO** KOH.I.NOOR HARDTMUTH 02/6972 CUSL 02/804967 DELTRON 02/2360015 ELEDRA 3S 02/349751 GBC 02/2041051, GBC 02/490335, GIGLIONI 02/2360397, GIGLIONI 02/654906, IL PUNTO EDP 02/666503 JENZI 02/808595 VEMA ELETTRONICA 02/6071793, MARCUCCI 02/7386051, SUPERGAMES 02/2716343, TEXAS INSTRUMENTS 02/804938 • **CINISELLO BALSAMO (MI)** GBC 02/6181801 • **MONZA (MI)** EURO SYSTEM INFORMATIKA 039/365038 • **S. GIULIANO MILANESE (MI)** PROFIT CENTER 02/9846081 • **PAVIA** MARUCCI 0382/22160 • **BOLZANO** ELECTRONIA 0741/26631 • **VERONA** BIANCHI 045/590011, MARCHIOTTO 045/32233 • **VICENZA** RCE 0444 35554 • **PADOVA** CALDIRONI 049/657544 CAPOVILLA 049/28998, ELEDRA 3S 049/655488 RCE 049 662364, RCE 049/44306 • **CITTADELLA (PD)** BERTO 049/590959 • **VENEZIA** CAPUTO 041/36836 • **MESTRE (VE)** PACINOTTI 041/57716 • **TREVISO** RCE 0422 262571 • **PORDENONE** BRISOTTO 0434/22613 • **UDINE** MOFERT 0432 294620 • **FELETTO UMBERTO (UD)** FANTON UDINE 0432/68091 • **TRIESTE** COMPUTER CENTRE 040/574090 • **PARMA** FABBRI MAURIZIO 0521 27093 BELLETTI 0521/21267 • **REGGIO EMILIA** IMPEL 0522/43745, COMPUTER CENTER 0522/31965 FICARELLI 0522 49604 • **MODENA** DELLA CASA 059 210385, IMPEL 059/225819 MARI MAURO 059 225096, MASETTI ELETTRONICA 059/219801, ORAM 059 216109 • **BOLOGNA** ELEDRA 3S 051 307781 GOZZI GIANNI 051/239635, MORINI E FEDERICI 051 520305 MINELLI 051 384547 BORSARI SARTI 051 861060 • **FERRARA** IL CENTROCONTABILE 0532 40350 • **RAVENNA** FRATELLI GAMBI 0544/22938 POLITECNICA BAGNOLI 0544 32364 • **FORLÌ** TELERADIO 0543 25009 • **CESENA (FO)** MIRRI ITALO 0547/24025 **PISTOIA** CDE 0573 31669 • **MONTECATINI TERME (PT)** NANNINI 0572 767801 • **AVENZA (MS)** CDE 0585/59006 • **VIAREGGIO (LU)** CDE 0584 49244 • **PISA** L'ELETTROGRAFICA 050 24220 • **FIRENZE** CASA DELLO SCONTO 055/363141 BRESCHI 055/262696 PAOLETTI FERRERO 055 294974, BIERRE 055/413113 • **SESTO FIORENTINO (FI)** CECCHUTI 055/442698 • **AREZZO** VIERI 0575/23227 • **SIENA** BROGI 0577 280956 • **LIVORNO** L'ELETTROGRAFICA 0586 808031 • **PIOMBINO (LI)** TOMI 0565/30270 • **GROSSETO** ELECTRONIC MARKET 0564/411090, ELIOGRAF 0564/21066 • **PERUGIA** M.T.E. 075 21269 COMER 075/74000 • **ANCONA** LANARI E PAOLETTI 071/82461 MARCHETTI 071/28252 • **JESI (AN)** SIGNORETTI 0731/52403 • **FALCONARA (AN)** AMBROGINI 071/910079 • **FABRIANO (AN)** CARDINALETTI 0732/3015 • **ASCOLI PICENO** NARDONI 0736/50068 • **FERMO (AP)** FOTO TONICI 0734/26137 • **MACERATA** DOMIZI 0733 49583 • **L'AQUILA** GRIMALDI O. 0862/20289 SEAD 0862/28585 • **ROMA** ELEDRA 3S 06 8127324 FBM 06/3960152, GR ELETTRONICA 06/3598112, TEMAX ITALIA 06/4952623, RADIOVITTORIA 06 3963642, CORTANI 06/461277 SUONOVIDEO 06 6374248, ELDO 06/4387061, ELDO 06/710493 ELDO 06/299394, ELDO 06/5560995, ELDO 06/8312706 ELDO 06/4745412, INFORMATIKA 06/837274, INGEGNERIA 2000 06/4744169, DIEMME 06/7853341, DE ANGELIS 06/4743983, NOVEL 06/6783424, TECNOMEC 06/573305 • **NAPOLI** CENTRO SANDOZ 081 314026, MERKEL 081/241866 SPOT 2 081/313634 • **SALERNO** MANZO 089 226894 • **BARI** AUDITORIUM 3 080/216106, TECNOSYSTEM 080/224954 DI PETTA 080/221634 RANIERI 080/211425 DISCORAMA 080/214983, SECI 080/366810 • **TARANTO** ELETTROJOLLY 099/325809, ZINGARELLI 099/26039 • **VIBO VALENTIA (CZ)** P/C.E.M E. 0963 44655 • **MESSINA** FOTO OTTICA RANDAZZO 090 772189 • **PALERMO** C.S.H. 091 290244, FOTO OTTICA RANDAZZO 091/585133 FOTO OTTICA RANDAZZO 091 569877 GB ELECTRONIC 091 260914, PICK-UP HIFIDELITY 091/296768, SPADARO 091 477463 • **CATANIA** FOTO OTTICA RANDAZZO 095 321533, OVSAR 095 316547 • **CAGLIARI** AZIENDA CARTARIA PISANO 070 288901 SIGEA 070 495644 •

Home Computer Texas Instruments.
Prezzo imbattibile. Software ineguagliabile.

Quando sceglierete un Home Computer Texas Instruments sceglierete un "vero" computer. Un computer che può crescere con voi e con la vostra famiglia. Un computer con cui potete giocare, inventare, studiare... insomma, un sistema che aiuta la vostra fantasia a crescere.

Tutto ciò è possibile grazie alla nostra gamma di software: dai videogiochi come gli Invaders, al calcio, agli scacchi, ai linguaggi di programmazione evoluti come l'"Editor Assembler" e l'"UCSD - PASCAL". Molti dei nostri programmi sono in forma di moduli di comando Solid State Software^{TM}, una esclusiva Texas Instruments. Per utilizzarli, dovete solo inserirli. E il gioco è fatto.

Se poi volete imparare a generare i vostri programmi, il TI-99/4A consente anche questo: il linguaggio TI-BASIC è immediatamente a vostra disposizione in consolle e il relativo manuale vi guiderà passo dopo passo nel mondo della programmazione. E quando vorrete ampliare il vostro sistema, potrete disporre di numerose periferiche quali l'espansione di memoria, l'RS232, il sistema di memoria a dischi, il sintetizzatore della voce ed altre ancora che, grazie al Peripheral Expansion System, unico nel suo genere, possono essere immediatamente e facilmente inserite ed utilizzate.

Vi sembra troppo? Provate l'Home Computer Texas Instruments dal rivenditore più vicino.

Per 499.000 lire (IVA esclusa), non troverete niente in grado di offrirvi tanto.

TEXAS INSTRUMENTS TI-99/4A
Caratteristiche tecniche

Microprocessore	TMS 9900 16 BIT
Grafica	16 colori, alta risoluzione
Linguaggi	TI-BASIC (disponibile in consolle) Extended Basic, UCSD - PASCAL TI-LOGO, Assembler
Memoria	Capacità di memoria interna disponibile all'utente 16 K RAM espandibile fino ad un massimo di 110 K ROM/RAM
Tastiera	Standard tipo macchina da scrivere
Software	1000 programmi tra cui scegliere in tutto il mondo
Capacità vocale Solid State	Sì

TM: marchio registrato Texas Instruments Inc.

TEXAS INSTRUMENTS

Comunicazione da PET a PET attraverso la "User Port"

di *J. Winn*

Se voi, o voi e un vostro amico, avete accesso a due PET, probabilmente avrete desiderato collegarli assieme e scambiare dati.

Il bus IEEE interno non si può usare,in quanto entrambi i PET sono *bus controller* ossia devono essere connessi solo con dispositivi *slave* cioè che eseguono gli ordini del *bus controller*.

Potreste acquistare molti tipi di dispositivi di ingresso/uscita: seriali, paralleli o di tipo *modem*, ma il sistema più semplice è usare la *user port* interna, ossia la porta parallela disponibile per l'utilizzatore del PET. Ecco come fare usando solo 12 fili e un semplice programma in Basic.

Per prima cosa diamo un'occhiata all'hardware necessario.

Il connettore della *user port* è posto sul retro del PET; guardando il PET da dietro, il connettore della porta parallela è quello centrale e i contatti sono posti nella fila inferiore e sono identificati dalle lettere dalla A alla N, con chiavi di polarizzazione tra i contatti A e B e tra L e M. A e N sono contatti di massa, da C a L ci sono i contatti della porta vera e propria (corrispondenti ai bit di una cella di memoria), mentre il contatto B corrisponde a CA1; esso viene usato per segnalare al PET la presenza di dati da memorizzare. Il contatto M, chiamato CB2, va collegato a CA1 del PET che riceve e controlla anch'esso il trasferimento dei dati. Per interconnettere i due PET dovrete collegare A con A, N con N, i contatti da C a L ai corrispondenti sul secondo PET, ma i fili B e M vanno incrociati: B del primo con M del secondo e viceversa. I cavi devono avere una lunghezza massima di sei metri onde evitare che i segnali in circolazione vengano disturbati.

Per controllare le linee di ingresso/uscita vengono usate alcune istruzioni PEEK e POKE. In ogni momento, un solo PET trasmette e l'altro riceve, anche se entrambi sono in grado di fare tutte e due le cose. Per spedire un dato, il "trasmettitore" per prima cosa attiva le otto linee dei dati, poi segnala attraverso CB2 che il dato è disponibile. Il "ricevitore", quando sente CA1 cambiare stato logico, carica il dato in memoria e segnala attraverso CB2 che ha ricevuto il byte ed è pronto per il prossimo.

Supponiamo di dover mandare un byte da un PET all'altro. Il programma 1 funziona da trasmettitore e il programma 2 da ricevitore.

La riga 20 mostra come in entrambi i programmi viene controllata la direzione dei dati. La riga 40 del programma che trasmette manda un byte (ASC(A$)) alle linee di uscita. Nel frattempo il ricevitore è bloccato sulla riga 40 e aspetta che il secondo bit della locazione di memoria 59469 vada a uno anziché rimanere a zero. Il segnale è mandato (da CB2 del trasmettitore a CA1 del ricevitore) tramite le righe 60 e 70. La riga 60 mette a uno i tre bit più significativi della locazione di memoria 59468 (lasciando gli altri bit inalte-

```
10 REM ****PROGRAMMA N.1****
15 REM PREPARA LA PORTA COME USCITA
20 POKE 59459,255
25 REM CHIEDE UN CARATTERE
30 INPUT"BATTI UN CARATTERE";A$
35 REM TRASMETTE IL CARATTERE
40 POKE 59457,ASC(A$)
50 REM SEGNALA AL RICEVITORE CHE IL DATO E' DISPONIBILE
60 POKE 59468,PEEK(59468) OR 224
70 POKE 59468,PEEK(59468) AND 31 OR 192
80 REM ASPETTA LA CONFERMA DAL RICEVITORE
90 IF (PEEK(59469) AND 2)<>2 THEN 90
100 GOTO 30
```

```
10 REM ****PROGRAMMA N.2****
15 REM PREPARA LA PORTA COME INGRESSO
20 POKE 59459,0
30 REM ASPETTA IL DATO
40 IF (PEEK(59469) AND 2)<>2 THEN 40
45 REM LEGGI IL DATO
50 D=PEEK(59457)
60 REM SEGNALA DI AVER RICEVUTO IL DATO
70 POKE 59468,PEEK(59468) OR 224
80 POKE 59468,PEEK(59468) AND 31 OR 192
85 REM SCRIVI IL CARATTERE RICEVUTO
90 PRINT CHR$(D)
100 GOTO 30
```

rati). La riga 70 rimette a zero il sesto bit e a uno il settimo e l'ottavo bit: questo fa cambiare stato per un attino a CB2 segnalando l'avvenuta trasmissione. La riga 90 blocca poi il trasmettitore fino a quando il ricevitore segnala di aver ricevuto il dato trasmesso. Il ricevitore fa questo con le righe 70 e 80 poi scrive il carattere ricevuto sullo schermo e aspetta il prossimo dato. Il trasmettitore riceve il segnale del ricevitore e richiede un altro carattere da trasmettere.

```
10 REM ****PROGRAMMA N.3****
12 REM TRASMETTITORE EVOLUTO (VERSIONE PER NUOVE ROM)
15 REM PREPARA LA PORTA COME INGRESSO
20 QQ=0: REM QQ DEVE ESSERE LA PRIMA VARIABILE
30 SQ=PEEK(42)+256*PEEK(43)+2: REM INDIRIZZO DI QQ NEL BUFFER
40 SH=152: REM INDIRIZZO DEL FLAG 'SHIFT'
45 REM LEGGI IL DATO
50 POKE 59459,255: REM PREPARA LA PORTA COME USCITA
60 REM MANDA IL CARATTERE DI SINCRONIZZAZIONE
70 D=ASC("%"): GOSUB 2000
80 PRINT"PRONTO A TRASMETTERE": PRINT"PREMI 'SHIFT' PER FERMARMI"
85 REM SCRIVI IL CARATTERE RICEVUTO
90 INPUT"SCRIVI UNA STRINGA";A$
95 REM TRASMETTE LA LUNGHEZZA DELLA STRINGA
100 QQ=LEN(A$): GOSUB 1000
110 FOR I=1 TO QQ
120 D=ASC(MID$(A$,I,1))
130 GOSUB 2000: REM MANDA UN CARATTERE ALLA VOLTA
140 NEXT
150 INPUT"QUANTI NUMERI A CASO";N
160 QQ=N: GOSUB 1000: REM TRASMETTE N
170 FOR I=1 TO N
180 QQ=RND(1)
190 GOSUB 1000: REM TRASMETTE IL NUMERO
200 NEXT
999 END
1000 REM TRASMETTE UN NUMERO IN VIRGOLA MOBILE
1010 FOR J=0 TO 4
1020 D=PEEK(SQ+J)
1030 GOSUB 2000: REM MANDA QQ UN BYTE ALLA VOLTA
1040 NEXT
1050 RETURN
2000 REM TRASMETTE UN BYTE
2010 POKE 59459,D: REM MANDA IL BYTE
2020 REM SEGNALA 'DATO PRONTO'
2030 POKE 59468,PEEK(59468) OR 224
2040 POKE 59468,PEEK(59468) AND 31 OR 192
2050 REM ASPETTA CONFERMA DAL RICEVITORE
2055 REM PERMETTE DI INTERROMPERE LA TRASMISSIONE
2060 IF ((PEEK(59469) AND 2)<>2) AND (PEEK(SH)<>1) THEN 2060
2070 IF PEEK(SH)=1 THEN 3000: REM SI FERMA
2080 RETURN
2990 REM SE INTERROTTO SI FERMA
3000 PRINT"TRASMISSIONE INTERROTTA": GOTO 999
```

La maggior parte delle applicazioni richiede di poter trasmettere più di un carattere alla volta. Per trasmettere un'intera stringa di caratteri o un numero in virgola mobile abbiamo bisogno di programmi più sofisticati ma basati su un identico principio.

Per trasmettere una stringa bisogna mandare per prima la lunghezza della stringa, dopo di che si può trasmetterla un carattere alla volta.

È importante la sincronizzazione tra trasmettitore e ricevitore

Per trasmettere un numero in virgola mobile, il modo più semplice sembra sia di usare una variabile Basic in una locazione di memoria conosciuta come viene fatto nel programma descritto più avanti.

Sorgono adesso altre due questioni. La prima è la sincronizzazione iniziale tra il trasmettitore e il ricevitore. Probabilmente il metodo migliore consiste nel mandare uno speciale carattere che funzioni da "preambolo" proprio per pulire il ricevitore da ogni dato casuale eventualmente presente. La seconda questione riguarda la possibilità di interrompere il programma se c'è qualcosa che non funziona: la linea 90 del trasmettitore e la linea 40 del ricevitore avrebbero potuto essere scritte utilizzando l'istruzione WAIT, ma non essendo possibile interrompere questo tipo di istruzione che togliendo la corrente ai circuiti, ciò sarebbe stato un po' pericoloso!

La maniera migliore di interrompere un programma senza usare il tasto STOP consiste nell'usare il tasto SHIFT nel modo descritto più sotto.

I programmi 3 e 4 sono una soluzione più elaborata per mandare stringhe o numeri in virgola mobile qualsiasi. Entrambi usano il tasto SHIFT per segnalare una sospensione del programma. (Con le vecchie ROM la locazione di memoria

516 è a zero se il tasto SHIFT non è premuto e a uno se invece è tenuto schiacciato. Con le nuove ROM questa locazione è la 152.) Il trasmettitore per prima cosa manda un % come preambolo per sincronizzare il ricevitore. Il carattere è scelto arbitrariamente ma dovrebbe essere il più possibile insolito nella trasmissione normale.

La variabile in virgola mobile, chiamata QQ in entrambi i programmi, *deve* essere la prima variabile definita nel programma. Questo perché così la sua posizione in memoria può essere trovata facilmente: all'inizio dell'area riservata alla memorizzazione delle variabili si trovano due byte per i due caratteri che formano il nome della variabile seguiti da cinque byte che rappresentano il vero e proprio numero in virgola mobile. L'area inizia a:

256*PEEK(43) + PEEK (42)
 per le nuove ROM
256*PEEK(125) + PEEK(124)
 per le ROM originali

Perciò la variabile SQ contiene la locazione, aumentata di due, dove inizia il numero da trasmettere.

I dati sono trasmessi o ricevuti tramite le subroutine a 1000 e 2000. A cominciare da 1000 c'è la subroutine che trasmette o riceve i numeri in virgola mobile (QQ); la subroutine in 2000 trasmette o riceve un byte alla volta (D).

Per interrompere la trasmissione o la ricezione bisogna che teniate premuto il tasto SHIFT fino a che il programma non salta alla subroutine 3000. Sia il trasmettitore che il ricevitore devono essere fermati separatamente, ma non ha importanza quale dei due viene fermato per primo.

La velocità di trasmissione è di circa dieci byte al secondo.

```
10 REM ****PROGRAMMA N.4****
12 REM RICEVITORE EVOLUTO (VERSIONE PER NUOVE ROM)
20 QQ=0: REM QQ DEVE ESSERE LA PRIMA VARIABILE
30 SQ=PEEK(42)+256*PEEK(43)+2: REM INDIRIZZO DI QQ NEL BUFFER
40 SH=152: REM INDIRIZZO DEL FLAG 'SHIFT'
50 POKE 59459,0: REM PREPARA LA PORTA COME INGRESSO
60 REM ASPETTA IL CARATTERE DI SINCRONIZZAZIONE
65 FOR I=1 TO 3: GOSUB 2000
70 IF D=ASC("%") THEN 80
72 NEXT
74 PRINT"SINCRONIZZAZIONE FALLITA": END
80 PRINT"PRONTO A RICEVERE": PRINT"PREMI 'SHIFT' PER FERMARMI"
90 GOSUB 1000: REM LEGGE LA LUNGHEZZA DELLA STRINGA
95 REM LEGGE UN BYTE ALLA VOLTA
100 A$="": FOR I=1 TO QQ: GOSUB 2000
110 A$=A$+CHR$(D): REM RICOSTRUISCE LA STRINGA
120 NEXT
130 PRINT"RICEVUTO ";A$
140 REM LEGGE QUANTI NUMERI VERRANNO TRASMESSI
150 GOSUB 1000: N=QQ
160 FOR I=1 TO N
170 GOSUB 1000: REM LEGGE I NUMERI
180 NEXT
999 END
1000 REM RICEVE UN NUMERO IN VIRGOLA MOBILE
1010 FOR J=0 TO 4
1020 GOSUB 2000: REM LEGGE UN BYTE ALLA VOLTA
1030 POKE SQ+J,D: REM RICOSTRUISCE QQ
1040 NEXT
1050 RETURN
2000 REM RICEVE UN BYTE
2010 REM ASPETTA IL DATO E PERMETTE L'INTERRUZIONE
2020 IF ((PEEK(59469) AND 2)<>2) AND (PEEK(SH)<>1) THEN 2020
2030 IF PEEK(SH)=1 THEN 3000: REM SI FERMA
2040 D=PEEK(59457): REM LEGGE IL BYTE
2050 REM SEGNALA 'DATO RICEVUTO'
2060 POKE 59468,PEEK(59468) OR 224
2070 POKE 59468,PEEK(59468) AND 31 OR 192
2080 RETURN
2990 REM SE INTERROTTO SI FERMA
3000 PRINT"RICEZIONE INTERROTTA": GOTO 999
```

Linguaggio macchina per trasferimenti più rapidi

Per ottenere ritmi di trasferimento più veloci bisogna ricorrere al linguaggio macchina: il programma 5 è una versione in linguaggio macchina dei programmi Basic 3 e 4, ma implementati in maniera leggermente differente.

La riga 10 abilita la variabile D% a ricevere singoli byte: ancora una volta *deve* essere la prima variabile definita nei programmi e le istruzioni PEEK devono essere modificate a 125 e 124 per le vecchie ROM. L'istruzione POKE 2,3 provvede al lincaggio per la funzione USR. La riga 150 scrive tramite l'istruzione POKE il codice macchina nel buffer per il secondo registratore a cassette. La riga 30 mette l'indirizzo di D% nel programma in linguaggio macchina e mette a zero D%. Le istruzioni DATA contengono il programma per le nuove ROM. Per adattare il programma alle ROM originali occorre cambiare i due 94 (alle righe 500 e 720) in due 176: essi indicano la localizzazione dell'accumulatore in virgola mobile usato dall'istruzione USR.

Per far trasmettere il programma, POKE 1,91 per terminare il lincaggio della funzione USR, poi mandare un byte di preambolo (ancora %) per sincronizzare la trasmissione. Per mandare singoli byte (riga 200); POKE il carattere nella locazione 832 e poi eseguire un SYS826. Per trasmettere un numero in virgola mobile (riga 220) passare il numero come argomento di USR; poiché USR deve essere posto uguale a qualcosa, può essere tranquillamente messo uguale alla variabile da trasmettere.

Ovviamente se un programma viene fatto trasmettere, l'altro deve essere messo in condizioni di ricevere: per prima cosa (riga 250) PO-

```
100 REM ****PROGRAMMA N.5****
110 REM VERSIONE PER NUOVE ROM
120 REM CON ROUTINE IN LINGUAGGIO MACCHINA
130 REM D% DEVE ESSERE LA PRIMA VARIABILE
140 D%=256*PEEK(43)+PEEK(42)+3: POKE 2.3
150 FOR I=826 TO 917: READ J: POKE I,J: NEXT
160 I=PEEK(D%): POKE 889.I: I=PEEK(D%-1): POKE 890.I: D%=0
170 REM ---TRASMETTE---
180 POKE 1,91: REM LINK PER FUNZIONE USR
190 POKE 832,ASC("%"): SYS 826: REM SINCRONIZZA
200 REM TRASMETTE UN BYTE ("A")
210 POKE 832,ASC("A"): SYS 826
220 REM TRASMETTE IL N. 1,23
230 X=1.23: X=USR(X)
240 REM ---RICEVE---
250 POKE 1,139: REM LINK PER FUNZIONE USR
260 FOR I=1 TO 3: SYS 873: IF D%=ASC("%") THEN 290
270 NEXT: REM CERCA IL PREAMBOLO
280 PRINT"SINCRONIZZAZIONE FALLITA": END
290 REM RICEVE UN BYTE
300 SYS 873: A$=A$+CHR$(D%): PRINTA$
310 REM RICEVE UN NUMERO
320 X=USR(0): PRINTX
330 REM DATA E CODICI MNEMONICI CORRISPONDENTI
340 REM
350 DATA 169,255      :'TBYTE LDA #$FF
360 DATA 141,67,232   :'      STA $E843
370 DATA 169,0        :'      LDA #$00
380 DATA 141,65,232   :'      STA $E841
390 DATA 173,76,232   :'      LDA $E84C
400 DATA 9,224        :'      ORA #$E0
410 DATA 141,76,232   :'      STA $E84C
420 DATA 41,31        :'      AND #$1F
430 DATA 9,192        :'      ORA #$C0
440 DATA 141,76,232   :'      STA $E84C
450 DATA 173,77,232   :'TWAIT LDA $E84D
460 DATA 41,2         :'      AND #$02
470 DATA 240,249      :'      BEQ TWAIT
480 DATA 96           :'      RTS
490 DATA 162,5        :'      LDX #$05
500 DATA 181,94       :'TFLPT LDA $5E,X
510 DATA 141,64,3     :'      STA $0340
520 DATA 32,058,03    :'      JSR TBYTE
530 DATA 202          :'      DEX
540 DATA 16,245       :'      BPL TFLPT
550 DATA 96           :'      RTS
560 DATA 169,0        :'RBYTE LDA #$00
570 DATA 141,67,232   :'      STA $E843
580 DATA 173,77,232   :'RWAIT LDA $E84D
590 DATA 41,2         :'      AND #$02
600 DATA 240,249      :'      BEQ RWAIT
610 DATA 174,65,232   :'      LDY $E84C
620 DATA 142,0,0      :'      STY $0000
630 DATA 173,76,232   :'      LDA $E84C
640 DATA 9,224        :'      ORA #$E0
650 DATA 141,76,232   :'      STA $E84C
660 DATA 41,2         :'      AND #$1F
670 DATA 9,192        :'      ORA #$C0
680 DATA 141,76,232   :'      STA $E84C
690 DATA 96           :'      RTS
700 DATA 162,5        :'      LDX #$05
710 DATA 32,069,03    :'RFLPT JSR RBYTE
720 DATA 148,94       :'      STY $5E,X
730 DATA 202          :'      DEX TBYTE
740 DATA 16,245       :'      BPL RFLPT
750 DATA 96           :'      RTS
```

KE 1,139 per terminare il lincaggio di USR. Poi aspettate che arrivi il carattere di preambolo e cautelatevi se non arriva (riga 280). Il ciclo FOR-NEXT a 260 e 270 non dovrebbe mai andare oltre I=2. Per ricevere singoli byte chiamate SYS873 e ricavate il dato dalla variabile D%. Per ricevere numeri, (riga 310) la variabile che deve contenere il numero deve essere posta uguale a USR. L'argomento della funzione USR non ha importanza in questo contesto.

Nella maggior parte dei programmi le righe da 180 a 190 e da 240 a 280 dovrebbero esere scritte come subroutine, da richiamare per cambiare un modo di funzionamento in un altro. Il maggior problema con questo programma è che non può essere fermato facilmente; la sincronizzazione, poi, deve essere perfetta, altrimenti uno dei due terminerà prima dell'altro, lasciando il secondo bloccato: uno o più SYS826 o SYS873 dal PET libero possono eventualmente sbloccare l'altro (il tipo di SYS usato dipende dal modo di funzionamento del PET bloccato).

Velocità di trasmissione

La velocità di trasmissione è buona: con un ciclo

FOR I=1 TO 2000: X=USR(I): NEXT

si impiegano circa 8.6 secondi per trasmettere 2000 numeri in virgola mobile ossia con un ritmo di 1395 byte/sec. Per trasferire i numeri in una matrice occorrono circa 12.5 secondi.

Per rilocare il programma, occorre cambiare i valori preceduti da uno zero nei DATA. Questi sono coppie che formano indirizzi: prima il meno significativo, poi il più significativo secondo la formula

BPS*256+BMS = IND.

Devono essere cambiati insieme alle istruzioni POKE in Basic e ai valori del lincaggio della funzione USR. Per gli utenti del DOS versione 4.0 si suggerisce di rilocare la routine per evitare che il sistema operativo usi il fondo del buffer della seconda cassetta.

Pixelator, generatore di caratteri per il VIC

di *J. Calloway*

La prima volta che capita di disegnare caratteri speciali per il VIC-20 Commodore, la cosa può anche sembrare divertente. Si tratta di preparare la carta con una matrice otto per otto e disegnare il carattere. Poi occorre convertire la figura in numeri come se le caselle scure fossero degli uno binari e quelle illuminate degli zero. Fatto questo, bisogna memorizzare questi numeri.

Poi, con un'istruzione POKE all'indirizzo 36869 la magia si compie. Lo schermo si riempie di strani segni. Ma, ecco! Al posto in cui dovrebbe esserci la A di READY c'è una nave spaziale, e al posto della D c'è un alieno con tre gambe!

Una volta che l'emozione è passata, tutto questo lavoro diventa molto noioso. Già è abbastanza fastidioso convertire il disegno in numeri ma inserirsi nelle istruzioni DATA è ancora più tediante e, oltretutto, soggetto a errori di battuta. Basta sbagliare una battuta e la nostra meravigliosa astronave sembra passata per una barriera di raggi laser ed è ridotta ad un colabrodo.

Progettare caratteri con Pixelator

Il programma Pixelator risparmia un po' della noia di questo lavoro e dà anche l'emozione di creare sempre nuovi caratteri. Pixelator dà contemporaneamente sullo schermo quattro aree di lavoro di otto per otto punti per creare, editare e confrontare diversi caratteri. Poi memorizza i caratteri così generati. Nel VIC standard con 3,5 K di memoria, Pixelator riesce a far stare fino a 64 caratteri diversi. Se disponete di memoria ausiliaria, il programma può gestire fino a 128 caratteri alla volta; esso può anche leggere i caratteri che sono contenuti nella memoria. Potete addirittura copiare i caratteri che sono già nelle ROM del VIC stesso. In seguito potete modificare i caratteri a vostro piacimento.

Come la maggior parte dei piccoli computer, il VIC contiene la mappa dei caratteri in ROM (indirizzi da 32768 a 36863). Diversamente dagli altri, il VIC, però, ha caratteri formati in una matrice otto per otto anziché cinque per sette. I punti che formano il carattere si chiamano *pixel* e sono i pezzi più piccoli dello schermo che possono essere controllati individualmente. Questo fatto rende i caratteri del VIC un po' strani da vedere (sono più larghi che alti), ma sfrutta completamente le possibilità della memoria. Per ogni carattere occorrono otto byte, e ogni byte corrisponde a una riga orizzontale del carattere. L'informazione verticale si ha dalla divisione del byte in zero e uno binari che corrispondono alle aree scure e luminose del carattere.

La memoria

Cambiando, con POKE, il numero contenuto all'indirizzo 36869, viene alterato il punto della memoria dove il Video Interface Chip (circuito integrato che controlla e gestisce il video del VIC e da cui proviene il nome stesso della macchina) cerca la mappa dei caratteri. Questo avviene automaticamente, per esempio, quando passate dal modo grafico al modo testo dalla tastiera. Il modo grafico corrisponde al valore 240 all'indirizzo citato prima, mentre il modo testo corrisponde al 242. Il valore intermedio 241, rappresenta i caratteri grafici inversi, ma quando si usano i caratteri grafici inversi da tastiera, il valore a 36869 non cambia.

Un valore di 252 sposta la mappa a 4096, ossia l'inizio della RAM standard da 3,5 K byte. Oltre il 252, l'indirizzo corrispondente viene incrementato di 1024 byte, fino a un massimo di 7168 che corrisponde al valore 255. A causa della lunghezza del programma, Pixelator usa il valore più alto, ossia 255.

Pixelator, una volta caricato e funzionante, occupa circa 3 K di memoria. Su VIC non espanso, restano liberi circa 500 byte che permettono di memorizzare 64 caratteri. Questo limite coincide con il fatto che la memoria sopra 7168 è occupata dalla mappa dello schermo.

Naturalmente, con la memoria

```
1 REM ********************************************
2 REM *                                          *
3 REM *              PIXELATOR                   *
4 REM *          VERSIONE VIC-20                 *
5 REM *                                          *
6 REM *         PERSONAL SOFTWARE                *
7 REM *                                          *
8 REM ********************************************
20 XX=7168:SC=7680:CL=38400
30 POKE51,240:POKE52,XX/256-1:POKE55,240
32 POKE56,XX/256-1
40 FORLX=16TO1STEP-1:READXZ:POKEXX-LX,XZ:NEXT
50 POKEXX-10,SC/256:POKEXX-1,XX/256-1
60 PRINT"▒▒▒";
70 FORY=1TO2:PRINT"▒▒▒?????????▒▒?????????"
80 FORZ=1TO8:PRINT"▒'▒          ▒'▒          ▒":NEXT
90 PRINT"▒▒'''''''''▒▒'''''''''▒":NEXT
100 POKE36879,25:F=0:J=0:SYSXX-16
102 PRINT"▒▒▒F1▒▒-CREA CARATTERI"
110 PRINT"▒▒F3▒▒-LEGGE MEMORIA"
120 GETS1$:IFS1$=""THEN120
130 IFS1$="▒"THENK=0:POKE36879,29:GOTO160
140 IFS1$="▒"THENPOKE36879,27:GOTO3500
150 GOTO120
160 IFJ=1THEN190
170 SYSXX-16:PRINT"▒SCEGLI"SPC(4)"F1 F3"
172 PRINT"SCHERMO:"SPC(2)"F5 F7";
180 GETS$:IFS$=""THEN180
190 IFASC(S$)>132THEN5100
200 GOTO180
210 VV=3:HH=1:F=88:GOTO250
220 VV=3:HH=11:F=109:GOTO250
230 VV=13:HH=1:F=462:GOTO250
240 VV=13:HH=11:F=483
250 POKEF+SC,160:IFK>0THENPOKEF+CL,3:GOTO270
260 IFJ=0THENPOKEF+CL,5:GOTO280
270 IFJ>0THENC=CJ:C0=CG:GOTO320
280 SYSXX-16:PRINT"▒SCEGLI IL CARATTERE";
290 GETC$:IFC$=""THEN290
300 GOSUB5000
310 IFCE=2ANDS2$="▒"THEN290
320 IFK=1ANDI=0ANDCE<>1THEN4000
330 IFCE>0THEN290
340 POKEF+SC,C:POKEF+CL,0:V=1:H=1
342 P=SC+23+VV*22+H:PA=P:PQ=PEEK(P)+72:PP=PQ
350 I=0:J=0:SYSXX-16:PRINT"▒F1-MEMORIZZA"
360 PRINT"F3-CANCELLA F5-RILOCA F7-MEMORIZZA/INCR";
370 GETG$:POKEP,PQ:POKEPA,PP:IFG$=""THEN370
380 IFASC(G$)=160THENPOKEP,ASC(G$):H=H+1:GOTO440
382 IFASC(G$)=32THENPOKEP,ASC(G$):H=H+1:GOTO440
390 IFG$="▒"THENV=V+1:GOTO440
400 IFG$="▒"THENV=V-1:GOTO440
410 IFG$="▒"THENH=H+1:GOTO440
420 IFG$="▒"THENH=H-1:GOTO440
430 IFASC(G$)<133ORASC(G$)>136THEN370
440 IFH>8THENH=1:V=V+1
```

espansa, tutto quello che dovete fare è scegliere una zona di memoria che non interferisca con la mappa dello schermo. Certe volte il problema viene risolto automaticamente, poiché la mappa del video viene spostata (così come la mappa del colore dei caratteri).

Le tre variabili in 20 vi permettono di cambiare il programma per compensare queste differenze. XX è la memoria per la mappa e deve sempre essere un multiplo intero di 1024. SC è la memoria per lo schermo e CL quella per il colore.

Quando fate girare il programma, per prima cosa vi viene proposta una scelta tra creare nuovi caratteri o leggerne di residenti in memoria. La scelta è codificata secondo il colore: verde o ciano rispettivamente. Se scegliete di creare caratteri, con il tasto F1, il bordo dello schermo cambia da bianco a verde e potete scegliere una delle quattro aree di lavoro con i tasti F1, F3, F5 e F7. Poi vi viene chiesto il carattere al cui indirizzo volete memorizzare il nuovo carattere.

Quattro opzioni

Una volta scelto il carattere, vedrete comparire un simbolo alto metà carattere all'angolo sinistro dell'area di lavoro scelta. Quello è il vostro cursore e voi, usando i tasti di controllo del cursore della tastiera, potete posizionarlo dove volete. Per disegnare un carattere usate la barra dello spazio. Con SHIFT e spazio lascerete una scia rossa. Senza lo SHIFT premuto, verrà cancellato quello che sta sotto al cursore. Per cancellare un'area di lavoro, basta che teniate premuto lo spazio finché tutto il rosso non sarà sparito.

Dopo aver elaborato il carattere a vostro piacimento, avete quattro possibilità. F1 memorizza gli otto byte nell'area di memoria appropriata e torna al formato di apertura. F3 abortisce il disegno che avete generato e torna all'apertura senza memorizzare il carattere. F5, invece, cambia nome al carattere, permettendovi di assegnargli un indirizzo diverso da quello scelto precedentemente. Questo modo è più utile quando state leggendo i caratteri dalla memoria, piuttosto

```
450 IFH<1THENH=8:V=V-1
460 IFV>8THENV=1:H=H+1
470 IFV<1THENV=8:H=H-1
480 PP=PEEK(P):PA=P:IFPP=104ORPP=232THENPP=PP-72
490 IFG$="■"THENK=0:POKEPA,PP:GOTO1000
500 IFG$="■"THENK=0:POKEPA,PP:GOTO100
510 IFG$="▌"THENI=1:POKEPA,PP:GOTO5200
520 IFG$="▌"THENJ=1:POKEPA,PP:GOTO1000
530 P=SC+(VV+V)*22+HH+H:PQ=PEEK(P)+72
540 GOTO370
1000 SYSXX-16:PRINT"▒MEMORIZZO ";:POKESC+10,C
1010 FORVE=1TO8:ZZ=0
1020 FORHY=1TO8:PO=SC+(VV+VE)*22+HH+HY
1030 IFPEEK(PO)=160THENZZ=ZZ+2↑(8-HY)
1040 NEXT
1050 POKEXX+C*8+VE-1,ZZ:NEXT:IFJ=0THEN100
2000 CJ=C+1:CG=C0+1:S$=CHR$(ASC(S$)+1)
2002 IFASC(S$)>136THENS$="■"
2010 IFCJ=64ANDXX=7168ANDSC=7680THENCE=2
2020 IFK=2THENK=1
2030 IFCG>127THENCG=0
2040 IFS2$="■"ANDCE=2THENJ=0:GOTO100
2050 IFK=0ANDCE=2THEN100
2060 GOTO190
3500 K=1:IFJ=1THEN3540
3510 SYSXX-16:PRINT"▒F1-CARICA DA RAM"
3520 PRINT"F2-ROMGRAF  ▒F4-REVERSE";
3522 PRINT"■F6-ROMTESTO ▒F8-REVERSE■";
3530 GETS2$:IFS2$=""THEN3530
3540 IFS2$="■"THENXR=XX:GOTO3580
3550 S2=ASC(S2$)-137
3552 IFS2>-1ANDS2<4THENXR=32768+1024*S2:GOTO3570
3560 GOTO3530
3570 IFS2>1THENPOKE36869,242:GOTO160
3580 POKE36869,240:GOTO160
4000 IFJ=0THENC0=C
4010 SYSXX-16
4012 PRINT"▒":PRINT"ECCO LA  ";S5$:POKE7713,C0
4020 FORD=1TO8:DA=PEEK(XR+C0*8+D-1):DI=0
4030 FORDD=1TO8:DI=INT(DA/2↑(8-DD))
4032 DA=DA-DI*2↑(8-DD)
4040 IFDI>0THENDO=160:GOTO4060
4050 DO=32
4060 IFDD=8ANDD<8THENZD=15:GOTO4090
4070 IFD=8ANDDD=8THENZD=-184:GOTO4090
4080 ZD=1
4090 ZF=SC+(VV+D)*22+HH+DD:POKEZF,DO
4092 POKEZF+ZD,PEEK(ZF+ZD)+72:NEXTDD,D
4100 IFCE>0THENK=2:GOTO4120
4110 GOTO340
4120 SYSXX-16:PRINT"▒RILOCA":GOTO290
5000 C=ASC(C$):CE=0
5010 ONINT(C/32)GOTO5060,5040,5050,5020,5040,5030
5020 CE=1:RETURN
5030 C=C-64
5040 C=C-32
5050 C=C-32
5060 IFJ=1THENC0=CG
```

che quando li create, ma funziona in tutti e due i modi. F7, per finire, memorizza il carattere e passa a quello successivo, senza costringervi a passare ogni volta per la sequenza “scelta dell’area di lavoro” e “scelta del carattere” da elaborare. Serve soprattutto per creare sequenze di caratteri. La procedura si ripete finché non si incontra il punto di domanda, che è l’ultimo dei 64 caratteri, dopo di che torna al formato di apertura.

Se, alla richiesta del formato di apertura, viene scelto il modo che legge i caratteri dalla memoria, il bordo diventa color ciano e vengono proposte cinque alternative. F1 legge da RAM; esso legge sia caratteri che avete già inserito in memoria sia i byte casuali presenti in memoria se non avete inserito nessun carattere fino al momento di leggere la memoria. F2 legge i caratteri dalle ROM del VIC nel modo grafico, mentre F4 carica i caratteri del modo grafico, ma in reverse. F6 e F8 servono per leggere i caratteri, normali o in reverse rispettivamente, del modo testo. Potete mescolare liberamente i caratteri prelevati con qualsiasi delle cinque possibili scelte e modificazioni secondo i nostri bisogni. Se per esempio vi serve l’intero alfabeto assieme ai vostri caratteri speciali, esiste una scorciatoia che consiste nel memorizzare i caratteri a 7168. Con POKE, in seguito, inserite 255 alla locazione 36869 e otterrete l’effetto di avere i caratteri normali, da @ a ?, con il tasto RVS ON e i vostri caratteri speciali con il tasto RVS OFF. Questa procedura funziona solo con il valore 255.

A questo punto il programma chiede quale area di lavoro deve essere utilizzata e che carattere, ma se selezionate un carattere grafico (oppure, nel modo testo, un carattere maiuscolo) dalla ROM, dovrete cambiargli nome con un carattere che sia inferiore a 64. Le scelte a vostra disposizione, a questo punto, sono le stesse viste prima: memorizzare, abortire, cambiare nome o memorizzare e incrementare il carattere. Se avete cambiato nome a un carattere, sia il carattere originale che il nuovo nome vengono incrementati dal tasto F7.

```
5070 IFXX=7168ANDC>63ANDSC=7680THENCE=2:RETURN
5080 RETURN
5100 ONASC(S$)-132GOTO210,220,230,240
5200 POKEF+CL,PEEK(36879)-24:POKEF+SC,160:GOTO4120
6000 DATA162,0,169,32,157,0,30,232
6010 DATA224,68,208,1,96,76,244,27
```

Listato 1. *Pixelator*

```
1 REM PIXAVER
10 XX=(PEEK(56)+1)*256
3000 SYSXX-16:PRINT"■PRIMO CARATTERE?";
3010 GETSR$:IFSR$=""THEN3010
3020 C$=SR$:GOSUB5000:SR=C:IFCE>0THEN3010
3030 PRINT"■"SPC(15)" "SR$;
3032 PRINTSPC(5)"ULTIMO CARATTERE?";
3040 GETLS$:IFLS$=""THEN3040
3050 C$=LS$:GOSUB5000:LS=C:IFCE=1THEN3040
3060 IFSR<=LSTHEN3070
3065 SS=SR:SR=LS:LS=SS:SS$=SR$:SR$=LS$:LS$=SS$
3070 SYSXX-16:PRINT"■SALVO DA "SR$" A "LS$;
3080 PRINT"■";:OPEN1,1,1,SR$
3090 SYSXX-16:PRINT"■SALVO DA "SR$" A "LS$
3100 PRINT#1,SR
3110 PRINT#1,LS
3120 FORCZ=SRTOLS
3130 FORLL=0TO7
3140 PRINT#1,PEEK(XX+CZ*8+LL)
3150 NEXTLL
3160 NEXTCZ
3170 CLOSE1
3180 END
5000 C=ASC(C$):CE=0
5010 ONINT(C/32)GOTO5060,5030,5040,5020,5030,5050
5020 CE=1:RETURN
5030 C=C-64:GOTO5060
5040 C=C-32:GOTO5060
5050 C=C-128
5060 IFXX=7168ANDPEEK(648)*256=7680ANDC>63THENCE=2
5070 RETURN
```

Listato 2. *Pixaver*

Salvate i vostri caratteri

Più che per il piacere di inventare, probabilmente vorrete creare i caratteri per usarli in qualche programma particolare, per esempio in qualche gioco sullo schermo. I tre programmi più piccoli che appaiono in questo articolo vi permettono di salvare i caratteri che avete creato con Pixelator. Per memorizzare i caratteri direttamente su cassetta, dovete fermare il programma con il tasto STOP e battere NEW per cancellare il programma dalla memoria.

Poi caricate il programma Pixaver. Esso vi permette di salvare su nastro un blocco di caratteri di qualsiasi lunghezza, fino a 64, in un solo file sequenziale di dati. Il primo dato del file rappresenta il codice usato dallo schermo per il primo carattere del blocco, mentre il secondo dato è il codice dell'ultimo carattere. Questo vi permette di salvare quanti blocchi volete; ogni file contiene le informazioni necessarie per memorizzare i dati al posto giusto. Ogni file, inoltre, ha il primo carattere del blocco come nome di riconoscimento. A questo punto potete anche spegnere il VIC.

Pixeloader, invece, legge i dati dal nastro in memoria. Notate che la riga 10 del programma stabilisce il valore di XX indirizzo di partenza della mappa di memoria. Cambiando questo valore è possibile memorizzare vari blocchi in diverse zone di memoria, aggirando in questo modo il limite di 64 caratteri che abbiamo incontrato con Pixelator. Assicuratevi che XX sia un multiplo intero di 1024, altrimenti i caratteri non corrisponderanno più ai caratteri della tastiera.

Il terzo programma accessorio si chiama Pixdata. Esso converte un blocco di caratteri in RAM in istruzioni DATA, una per ogni carattere. I numeri di riga delle istruzioni DATA corrispondono al codice del carattere sommato a 5000. Le istruzioni DATA sono un modo molto inefficente di usare la memoria, ma sono più convenienti della registrazione su nastro perché possono essere inserite all'interno di un programma risparmiando le operazioni necessarie per leggere i caratteri dalla cassetta. Pixdata non è stato progettato per essere orientato all'utilizzatore come gli altri programma visti, perché è stato ridotto all'osso. Probabilmente dovrete modificare alcune righe del programma per adattarlo alle vostre esigenze ogni volta che utilizzerete il programma. I valori di SR e LS, inizializzati nella riga 30, per esempio, rappresentano i codici del primo e dell'ultimo carattere, rispettivamente. Se avete solo 3,5 K di memoria, non è possibile creare più di 30 caratteri alla volta senza incorrere in un OUT OF MEMORY ERROR.

Quello che rende Pixdata interessante è il fatto che si autodistrugge dopo aver terminato il suo compito, risparmiandovi di cancellarlo riga per riga per fare posto al vostro programma. (Ricordatevi di farne una copia su nastro prima di farlo girare!)

```
1 REM PIXLOADER
10 XX=7168
20 OPEN1,1,0
30 INPUT#1,SR
40 INPUT#1,LS
50 FORS=SRTOLS
60 FORRR=0TO7
70 INPUT#1,C:POKEXX+S*8+RR,C:NEXTRR:NEXTS
```

Listato 3. *Pixeloader*

Il segreto di Pixdata sta nel modo in cui il VIC memorizza le righe di programma Basic. I primi due byte di ogni riga sono il numero di riga della prossima istruzione. Il terzo e il quarto byte sono il numero di riga dell'istruzione. Dopo questi quattro byte ci sono dei dati che possono rappresentare o comandi Basic abbreviati (in inglese *token*), o caratteri ASCII. Tutti i numeri sono trattati come stringhe ASCII, perciò un'istruzione DATA necessita di tre byte per ogni singolo valore. Il numero 128, per esempio, diventa 49, 50, 56.

Aggiungete un 44 per ogni virgola e capirete perché una istruzione DATA usa fino a quattro volte più memoria del numero che rappresenta.

Pixdata comincia a creare le DATA dalla riga 5120, valore in ZZ nella riga 40. La riga 10 pone anch'essa a 5120 il limite massimo per la memoria dedicata alle righe Basic, proteggendo così le istruzioni DATA create dallo stesso programma. Quando il programma ha terminato di generare le DATA, esegue un POKE del valore di ZZ nei primi due byte della riga 1, che contiene "REM RIGA DA CANCELLARE DOPO AVER TERMINATO"; in questo modo l'interprete Basic salta direttamente dalla riga 1 alla 120, prima riga delle istruzioni DATA e in questo processo distrugge Pixdata. Quando poi voi cancellate la riga 1 scrivendo 1 su una riga libera dello schermo e poi battendo RETURN, l'interprete compatta le righe con le DATA all'inizio della memoria. Se avete aggiunto RAM al vostro VIC, assicuratevi che i due indirizzi nella riga 170 corrispondano prima di far girare il programma.

Per utilizzare le istruzioni DATA in un programma, avrete bisogno di una riga come:

FOR L=SR TO LS: FOR M=0 TO 7: READ C: POKE XX+L*8+M,C: NEXT M: NEXT L

Il valore di XX (mappa di memoria), di SR (primo carattere) e di LS (ultimo carattere) devono essere gli stessi usati in Pixdata.

I listati di questi quattro programmi sono stati redatti con delle norme diverse da quelle solitamente usate per gli altri listati della rivista. Questo è stato fatto per permettere anche a chi non possiede l'espansione di memoria per il VIC di far uso dei programmi. I caratteri speciali che appaiono nei listati sono quelli corrispondenti ai caratteri che il VIC stampa sul video. La stampa è stata effettuata con la stampante originale del VIC. Ci scusiamo perciò per la qualità della stampa e per la scarsa leggibilità dovuta alla mancanza di spazi tra le istruzioni, ma lo scopo vale la pena. I listati vanno battuti come sono, saltando le istruzioni REM. Questo discorso non vale, invece, per il quarto programma, Pixdata, che deve essere battuto così com'è, compresi i REM.

```
1 REM RIGA DA CANCELLARE DOPO AVER TERMINATO
5 REM DEVE UGUAGLIARE ZZ
9 REM SALVARE SU NASTRO PRIMA DI ESEGUIRE
10 POKE51,0:POKE52,20:POKE55,0:POKE56,20
20 XX=7168:S=26
29 REM PRIMO E ULTIMO CARATTERE
30 SR=0:LS=26
40 ZZ=5120:AA=ZZ
50 POKEZZ-1,0
60 FORL=SRTOLS
70 L2=INT((L*10+5000)/256)
71 L1=(L*10+5000)-L2*256
72 POKEZZ+2,L1:POKEZZ+3,L2
80 POKEZZ+4,131:X=4
90 FORLL=0TO7
100 S$=STR$(PEEK(XX+L*8+LL):S=LEN(S$)
110 FORLZ=2TOS:X=X+1
112 POKEZZ+X,ASC(MID$(S$,LZ,1)):NEXT
120 IFLL=7THEN140
130 X=X+1:POKEZZ+X,44:NEXT
140 X=X+1:POKEZZ+X,0
150 X=X+1:Z2=INT((ZZ+X)/256)
155 Z1=ZZ+X-Z2*256
160 POKEZZ,0:POKEZZ+1,0
170 A2=INT(AA/256):A1=AA-A2*256
175 POKE4097,A1
180 POKE4098,A2 POKE65,30
```

Listato 4. *Pixdata*

La carta del cielo

Computer: Apple II	**Autore:** Peter Dennings
Modello: Europlus	**Versione italiana:** Mauro Boscarol
Configurazione: 48 K	**Note:** Disponibile su dischetto

Un oroscopo è una mappa del cielo nel momento della nascita di un individuo. Esso illustra le posizioni dei corpi celesti in relazione alle dodici case e ai dodici segni attraverso i quali passano. Queste posizioni nel momento della nascita si pensa siano di grande importanza nella vita dei singoli individui. La mappa celeste di queste posizioni costituisce l'oroscopo di quella persona e viene detta "la carta del cielo".

Questo programma calcola e visualizza la "carta". Usando il sistema Koch di suddivisione delle case, il programma produce la carta del cielo per ogni data che sceglierete. Ciò vi farà risparmiare tempo ed energia eliminando il laborioso processo di creazione della carta. L'interpretazione degli aspetti a questo punto è lasciata a voi.

Uso

Per usare il programma dovete introdurre i dati standard di nascita usati dagli astrologi. Dapprima battete T o S, secondo che desiderate un oroscopo normale (T per tropicale) o corretto del movimento di precessione degli equinozi (S per siderale). Il programma quindi vi chiede la data di nascita, l'ora, il fuso orario, la longitudine e la latitudine. Una volta introdotti questi dati, viene tracciata la "carta" sullo schermo e vengono visualizzati gli "aspetti" (cioè le relazioni angolari di un pianeta con gli altri) dei corpi celesti. Ciò risparmia agli astrologi il pesante lavoro di ricavare gli aspetti da tavole di riferimento per tracciare la "carta" di una certa nascita.

Per introdurre i dati, usare il seguente formato:

Data: GG/MM/AAAA –return–

Il giorno, il mese e l'anno di nascita. Il giorno è un numero da 01 a 31, il mese è un numero da 01 a 12 e l'anno può andare da 1900 a 2000. Le sbarre (/) servono per separare giorno, mese e anno.

I giorni e i mesi di una sola cifra devono avere lo zero (0) davanti.

AM o PM –return–

AM significa che la nascita è avvenuta tra mezzanotte e mezzogiorno, PM tra mezzogiorno e mezzanotte.

Ora: HH.MM –return–

L'ora si introduce in ore e minuti. HH può andare da 00 a 11, dove 00 è uguale a 12 (12 e 03 deve essere battuto come 00.03) e MM può andare da 00 a 59. Ore e minuti sono separati da un punto. Se nel giorno di nascita era in vigore l'ora legale, bisogna sottrarre un'ora (vedi tabella).

Fuso orario: H –return–

Il fuso orario indica il numero di ore dal tempo medio di Greenwich dove H è uguale a zero.

Per zone a est di Greenwich usare il segno meno (−) prima del numero di ore. Per esempio, l'Italia è 1 ora a est di Greenwich e deve essere indicata con −1.

Longitudine: GG.MM –return–

La longitudine è introdotta nel formato GG.MM dove GG sono i gradi e MM i minuti. Le longitudini ovest sono rappresentate da un numero positivo, quelle a est da un numero negativo. Per esempio la longitudine 76 gradi e 18 minuti ovest deve essere battuta 76.18. Tutte le località italiane hanno longitudine negativa.

Latitudine: GG.MM –return–

Per la latitudine si usa lo stesso formato della longitudine.

Le latitudini nord (sopra l'equatore) sono rappresentate da numeri positivi, le latitudini sud da numeri negativi. Tutte le località italiane hanno latitudine positiva. Si veda la tabella delle coordinate geografiche delle principali città italiane.

Questo programma non riconosce gli errori, quindi prima di premere RETURN assicuratevi di aver impostato dei dati corretti.

Dopo aver introdotto tutti i dati, il programma produce la "carta" di quella nascita. Per vedere gli

aspetti planetari battete A e le relazioni verranno visualizzate al centro dello schermo.

Per introdurre nuovi dati e produrre una nuova carta battete N e procedete come descritto sopra.

In figura 1 è riprodotta una "carta" in bianco che potete ricopiare per annotare le posizioni dei pianeti e i risultati del programma.

ORE LEGALI IN ITALIA

Anno	Inizio		Fine	
1916	3/6	(ore 24)	30/9	(ore 24)
1917	31/3	(ore 24)	30/9	(ore 24)
1918	9/3	(ore 24)	6/10	(ore 24)
1919	1/3	(ore 24)	4/10	(ore 24)
1920	20/3	(ore 24)	18/9	(ore 24)
1940	14/6	(ore 24)	31/12	(ore 24)
1941	1/1	(ore 0)	31/12	(ore 24)
1942	1/1	(ore 0)	2/11	(ore 3)
1943	20/3	(ore 2)	4/10	(ore 3)
1944	3/4	(ore 2)	2/10	(ore 3)
1945	2/4	(ore 2)	16/9	(ore 24)
1946	17/3	(ore 2)	6/10	(ore 3)
1947	16/3	(ore 0)	5/10	(ore 1)
1948	29/2	(ore 2)	3/10	(ore 3)
1966	22/5	(ore 0)	24/9	(ore 24)
1967	28/5	(ore 0)	23/9	(ore 24)
1968	26/5	(ore 0)	21/9	(ore 24)
1969	1/6	(ore 0)	27/9	(ore 24)
1970	31/5	(ore 0)	27/9	(ore 0)
1971	23/5	(ore 0)	26/9	(ore 1)
1972	28/5	(ore 0)	1/10	(ore 0)
1973	3/6	(ore 0)	29/9	(ore 1)
1974	26/5	(ore 0)	29/9	(ore 1)
1975	1/6	(ore 0)	28/9	(ore 0)
1976	30/5	(ore 0)	25/9	(ore 24)
1977	22/5	(ore 0)	24/9	(ore 24)
1978	28/5	(ore 0)	1/10	(ore 0)
1979	27/5	(ore 0)	30/9	(ore 0)
1980	6/4	(ore 2)	28/9	(ore 3 legali)
1981	29/3	(ore 2)	17/9	(ore 3 legali)
1982	28/3	(ore 2)	26/9	(ore 3 legali)

ITALIA

Località	Long.	Lat.
Agrigento	−13.55	37.18
Alessandria	−8.38	44.54
Ancona	−13.31	43.37
Aosta	−7.20	45.44
Aquila (L')	−13.24	42.34
Arezzo	−11.53	43.28
Ascoli Piceno	−13.36	42.52
Assisi	−12.38	43.05
Asti	−8.12	44.54
Avellino	−14.48	40.54
Bari	−16.53	41.07
Belluno	−12.13	46.08
Benevento	−14.48	41.07
Bergamo	−9.39	45.42
Bologna	−11.20	44.30
Bolzano	−11.20	46.30
Brescia	−10.12	45.32
Brindisi	−17.56	40.39
Cagliari	−9.06	39.13
Caltanissetta	−14.04	37.29
Campobasso	−14.40	41.34
Capri	−14.15	40.32
Carrara	−10.06	44.05
Caserta	−13.50	41.30
Catania	−15.05	37.30
Catanzaro	−16.35	38.54
Chieti	−14.11	42.21
Civitavecchia	−11.47	42.05
Como	−9.05	45.49
Cosenza	−16.15	39.17
Cremona	−9.01	45.05
Crotone	−17.08	39.05
Cuneo	−7.33	44.23
Enna	−14.17	37.33
Faenza	−11.53	44.17
Ferrara	−11.38	44.50
Firenze	−11.16	43.46
Fiume	−14.27	45.20
Foggia	−15.34	41.28
Forlì	−12.02	44.13
Frosinone	−13.22	41.30
Genova	−8.55	44.25
Gorizia	−13.37	45.57
Grosseto	−11.07	42.46
Iglesias	−9.32	39.18
Imperia	−8.02	43.54
La Spezia	−9.49	44.06
Lecce	−18.10	40.21
Lipari	−14.57	38.28
Livorno	−10.19	43.33
Lodi	−9.30	45.18
Lucca	−10.30	45.50
Mantova	−10.47	45.09
Marsala	−12.26	37.48
Matera	−16.37	40.41
Merano	−11.10	46.41
Messina	−15.33	38.11
Milano	−9.12	45.28
Modena	−10.56	44.38
Monza	−9.15	45.35
Napoli	−14.16	40.52
Novara	−8.36	45.27
Nuoro	−9.20	40.20
Padova	−11.52	46.23
Palermo	−13.22	38.07
Parma	−10.10	44.48
Pavia	−9.09	45.11
Perugia	−12.23	43.07
Pesaro	−12.55	43.55
Pescara	−14.13	42.28
Piacenza	−9.42	45.03

continua nella prossima pagina

ITALIA

Località	Long.	Lat.
Pisa	−10.24	43.43
Pistoia	−10.55	43.56
Pola	−13.51	44.52
Potenza	−15.49	40.38
Rapallo	−9.14	44.20
Ravenna	−12.12	44.25
Reggio Calabria	−15.39	38.06
Reggio Emilia	−10.38	44.41
Rimini	−12.35	44.03
Roma	−12.29	41.54
Salerno	−14.46	40.41
Sanremo	−7.47	43.49
Sassari	−8.33	40.44
Savona	−8.29	44.18
Sestri Levante	−9.23	44.16
Siena	−11.18	43.19
Siracusa	−15.17	37.04
Sorrento	−14.23	40.37

ITALIA

Località	Long.	Lat.
Taranto	−17.15	40.28
Teramo	−13.43	42.39
Terni	−12.39	42.34
Tivoli	−12.48	41.58
Torino	−7.42	45.04
Trapani	−12.32	38.01
Trento	−11.08	46.03
Treviso	−12.15	45.40
Trieste	−13.46	45.39
Udine	−13.14	46.03
Urbino	−12.38	43.44
Varese	−8.49	45.50
Venezia	−12.20	45.26
Verona	−11.32	45.32
Vicenza	−11.32	45.32
Viterbo	−12.07	42.25
Voghera	−9.01	44.59

```
1 REM *******************************************
2 REM *                                         *
3 REM *              ASTROLOGIA                 *
4 REM *                                         *
5 REM *         VERSIONE APPLE II               *
6 REM *                                         *
7 REM *         PERSONAL SOFTWARE               *
8 REM *                                         *
9 REM *******************************************
10  DATA  AR,SD,TO,ME,GE,VE,CA,MA,LE,GI,VE,
     SA,BI,UR,SC,NE,SA,PL,CP,LU,AQ,NO,PE,0
15  DATA   1,13,1,19,11,19,21,19,31,19,31,13
     ,31,7,31,1,21,1,11,1,1,1,1,7
20  DATA   CON07000,OPP07180,TRI07120,QUA070
     90,SES05060,SQU02045,SSE02135
22  DATA  QUI02150
30  DATA 358.47584,35999.0498,-.00015,.01675
     1,-.41E-4,0,1.00000013,101.22083
32  DATA 1.71918,.00045,0,0,0,0,0,0,102.2793
     8,149472.515,0,.205614,.2E-4,0
34  DATA .387098,28.75375,.37028,.00012,47.1
     4594,1.1852,.00017,7.00288,.00186
36  DATA -.1E-4,212.60322,58517.8039,.00129,
     .00682,-.4E-4,0,.723332
38  DATA 54.38419,.50819,-.00139,75.77965,.8
     9985,.00041,3.39363,.001,0,319.5293
40  DATA 19139.8585,.00018,.09331,.9E-4,0,1.
     52369,285.43176,1.06977,.00013
42  DATA 48.78644,.77099,0,1.85033,-.00068,.
     1E-4,225.32833,3034.69202
44  DATA -.00072,.04833,.00016,0,5.202561,27
     3.27754,.59943,.0007,99.44378
46  DATA 1.01053,.00035,1.30874,-.005696,0,1
     75.46622,1221.55147,-.0005
48  DATA .05589,-.00035,0,9.55475,338.30777,
     1.08522,.00098,112.79039,.873195
50  DATA -.00015,2.49252,-.00392,-.2E-4,72.6
     4882,428.37911,.8E-4,.046344
52  DATA -.3E-4,0,19.21814,98.07155,.98577,-
     .00107,73.4771,.49867,.00131
54  DATA .77246,.00063,.4E-4,77.73067,18.46
     134,-.7E-4,.008997,0,0
56  DATA 30.10957,276.04597,.32564,.00014,13
     0.68136,1.09894,.000249,1.77924
58  DATA -.00954,0,229.94722,144.91306,0,.24
     864,0,0,39.51774,113.52139,0,0
60  DATA 108.95444,1.39601,.00031,17.14678,0
     ,0
115  DEF  FN S(X) =  SIN (X * PI / 180): DEF
      FN D(X) = X * 180 / PI:T$ = "S0000000O
     QQQQQQQQQ"
116  HOME :PI = 3.14159265
120  DEF  FN Q(X) =  SGN (X) * ( INT ( ABS (
     X)) + ( ABS (X) -  INT ( ABS (X))) * 10
     0 / 60):V$ = "SQQQQQQQ"
125  DEF  FN U(X) = X - ( INT (X / MO) * MO)
     : DEF  FN W(X) =  INT (X * 100 + .5) /
     100: DEF  FN R(X) = X * PI / 180
130  DEF  FN Y(X) =  ATN ( SQR (1 - X * X) /
     X)
135  DIM H$(12),R(12,2),H(13),Z$(12),C$(12),
     A$(8),K(12),C(12):K$ = "              "
136  VTAB (8): HTAB (10)
137  INVERSE : PRINT "T";: NORMAL : PRINT "R
     OPICALE ";: INVERSE : PRINT "S";: NORMAL
     : PRINT "IDERALE": GOSUB 5000:SD = 0: IF
     G$ = "S" THEN SD = 1
138  HOME
140  DEF  FN X(X) =  ATN (X /  SQR (1 - X *
     X)):MO = 360: FOR I = 1 TO 12: READ Z$(
     I),C$(I): NEXT I
145  FOR I = 1 TO 12: FOR J = 1 TO 2: READ R
     (I,J): NEXT J: NEXT I: FOR I = 1 TO 8: READ
     A$(I): NEXT I
147  VTAB 8
150  PRINT  TAB( 10);: INPUT "DATA(GG MM AAA
     A)=";A$:D =  VAL ( MID$ (A$,1,2))
155 Y =  VAL ( MID$ (A$,7,5)):M =  VAL ( MID$
     (A$,4,2)): PRINT  TAB( 10);: INPUT "AM
     PM=";F$
160  PRINT  TAB( 10);: INPUT "ORA(HH.MM)=";F
     :FO = F: PRINT  TAB( 10);: INPUT "FUSO
     ORARIO=";X:XO = X:F =  FN Q(F) + X
165  PRINT  TAB( 10);: INPUT "LONGITUDINE GG
     .MM)=";L5:LO5 = L5:L5 =  FN Q(L5): IF F
     $ = "PM" THEN F = F + 12
170  PRINT  TAB( 10);: INPUT "LATITUDINE(GG.
     MM)=";B4:BO4 = B4:B4 =  FN R( FN Q(B4))
     :X = 0
171 LA = B4
172  FOR ZO = 8 TO 15: HTAB (10): VTAB (ZO):
      PRINT "                              ": NEXT
     ZO
173  HOME
174  VTAB (8): PRINT  TAB( 13);"DATA=";D;"/"
     ;M;"/";Y: PRINT  TAB( 13);"ORA ";FO;F$:
      PRINT  TAB( 13);"FO=";XO: PRINT  TAB(
     13);"LONG=";LO5: PRINT  TAB( 13);"LAT="
     ;BO5
175 IM = 12 * (Y + 4800) + M - 3:J = (2 * (I
     M -  INT (IM / 12) * 12) + 7 + 365 * IM
     ) / 12
177 J =  INT (J) + D +  INT (IM / 48) - 3208
     3: IF J    = 2299171 THEN 180
```

```
178 J = J +  INT (IM / 4800) -  INT (IM / 12
     00) + 38
180 T = ((J - 2415020) + F / 24 - .5) / 3652
     5
182 C(11) =  FN U((933060 - 6962911 * T + 7.
     5 * T * T) / 3600): IF SD = 1 THEN  GOSUB
     700
185 RA =  FN R( FN U((6.646065556 + 2400.051
     262 * T + 2.5805E - 5 * T * T + F) * 15
     - L5))
195 OB =  FN R(23.45229444 - .0130125 * T)
200 MC =  ATN ( TAN (RA) /  COS (OB)): IF MC
     < 0 THEN MC = MC + PI
202  IF  SIN (RA) < 0 THEN MC = MC + PI
204 MC =  FN D(MC)
207 X =  FN D(RA) / 15:Y =  INT (X):Z = (X -
     Y) * 60:X =  INT ((Z -  INT (Z)) * 60):
     Z =  INT (Z)
210  PRINT  TAB( 10);"   TS=";  STR$ (Y) + "H
     " +  STR$ (Z) + "M" +  STR$ (X) + "S": GOSUB
     9000
211  GOSUB 430
212  NORMAL
220  FOR I = 1 TO 9:MO = 2 * PI: GOSUB 395:M
      =  FN U(S): GOSUB 395:E =  FN D(S):EA =
     M: FOR A = 1 TO 5
225 EA = M + E *  SIN (EA): NEXT A: READ AU:
     X = AU * ( COS (EA) - E):Y = AU *  SIN
     (EA) *  SQR (1 - E * E)
230  GOSUB 410: GOSUB 395:A = A + S: GOSUB 4
     00:D = X:X = Y:Y = 0: GOSUB 410: GOSUB
     395:AN = S
235  GOSUB 395:A = A + S: GOSUB 400:ZZ = Y:Y
      = X:X = D: GOSUB 410:A = A + AN: IF A <
     0 THEN A = A + 2 * PI
240  GOSUB 400:XX = X:YY = Y: GOSUB 305
245  IF I = 1 THEN C1 = K:C =  FN D( FN U(K +
     PI)):M1 = R:X1 = XX:Y1 = YY:Z1 = ZZ:X =
     1: GOTO 260
250 W1 = (((R ^ .5 + M1 ^ .5) * (M1 ^ .5 * R
     .5)) / (R ^ 1.5 + M1 ^ 1.5)) -  COS
     (C1 - K)
255 XX = XX - X1:YY = YY - Y1:ZZ = ZZ - Z1: GOSUB
     305:X = 1: IF W1 < 0 THEN X =  - X
260 MO = 360:C(I) =  FN U(C + SD) * X: NEXT
     I
265 LL = 973563 + 1732564379 * T - 4 * T * T
     :G = 1012395 + 6189 * T:G1 = 1290584 +
     146485227 * T - 72 * T * T
270 D = 1262655 + 1602961611 * T - 5 * T * T
     :M = 3600:L = (LL - G1) / M:L1 = ((LL -
     D) - G) / M
275 F = (LL - C(11) * M) / M:D = D / M:ML =
     22639.6 *  FN S(L1) - 669 *  FN S(L1)
280 Y = 2 * D:ML = ML - 4586.4 *  FN S(L - Y
     ) + 2369.9 *  FN S(Y) + 769 *  FN S(2 *
     L) - 706 *  FN S(L + L1 - Y)
285 ML = ML - 411.6 *  FN S(2 * F) - 212 *  FN
     S(2 * L - Y) + 192 *  FN S(L + Y) - 165
     *  FN S(L1 - Y) [illegible]
290 ML = (LL + ML - 165 *  FN S(L1 - Y) + 14
     8 *  FN S( [illegible] L1) - 110 *  FN S(L + L1)
     - 55 *  FN S(2 * F - Y)) / M
295 C(10) =  FN U(ML + SD)
300 C(11) =  FN U(C(11) + SD): GOTO 485
305 X = XX:Y = YY: GOSUB 410:K = A:C =  FN D
     (A): RETURN
310 ZZ =  INT (Z):C =  INT (ZZ / 30) + 1:Z1 =
      INT ( FN W((ZZ / 30 -  INT (ZZ / 30)) *
     30))
315 X$ =  STR$ (Z1): IF Z1 < 10 THEN X$ = "0
     " +  RIGHT$ (X$,1)
320 ZZ$ =  RIGHT$ ( STR$ ( INT ( [illegible]
     0 + .5)),2)
325  IF  VAL (ZZ$) < 10 THEN ZZ$ = "0" +  RIGHT$
     (ZZ$,1)
330 Z$ =  RIGHT$ (X$,2) + Z$(C) + ZZ$: RETURN


395  READ S,S1,S2:S =  FN R(S + S1 * T + S2 *
     T * T): RETURN
400  IF A = 0 THEN A = 1.75E - 9
405 X = R *  COS (A):Y = R *  SIN (A): RETURN
410  IF Y = 0 THEN Y = 1.75E - 9
415 R = (X * X + Y * Y) ^ .5:A =  ATN (Y / X
     ): IF A < 0 THEN A = A + PI
420  IF Y < 0 THEN A = A + PI
425  RETURN
430  VTAB 1: FOR I = 1 TO 5: NORMAL
435  PRINT K$;: INVERSE : PRINT  MID$ (H$(11
     ),I,1);: NORMAL : PRINT K$;: INVERSE : PRINT
      MID$ (H$(10),I,1);: NORMAL : PRINT K$;
440  INVERSE : PRINT  MID$ (H$(9),I,1): NEXT
     I: PRINT " "H$(12)"   " MID$ (H$(11),6);

445  PRINT K$ MID$ (H$(10),6,1)K$ MID$ (H$(9
     ),6,1)"   "H$(8)" "
446  INVERSE
450  FOR I = 1 TO 5: HTAB 10: PRINT " ";: HTAB
     30: PRINT " ": NEXT I
455  PRINT " "H$(1)"     ";: HTAB 30: PRINT "
       "H$(7)" "
460  FOR I = 1 TO 5: HTAB (10): PRINT " ";: HTAB
     30: PRINT " ": NEXT I
465  PRINT " "H$(2)"   " MID$ (H$(3),1,1)" CA
     SE DI " MID$ (H$(4),1,1);
470  PRINT " PLACIDUS" MID$ (H$(5),1,1)"   "H
     $(6)" ": FOR I = 1 TO 5: NORMAL : PRINT
     K$;
475  INVERSE : PRINT  MID$ (H$(3),I + 1,1);:
      NORMAL : PRINT K$;: INVERSE : PRINT  MID$
     (H$(4),I + 1,1);: NORMAL : PRINT K$;
480  INVERSE : PRINT  MID$ (H$(5),I + 1,1): NEXT
     I: RETURN
485  FOR I = 1 TO 11
490 K(I) =  ABS (C(I)): NEXT I: FOR I = 1 TO
     11 - 1: FOR J = I + 1 TO 11: IF K(J)
      = K(I) GOTO 500
495 K1 = K(I):K(I) = K(J):K(J) = K1
500  NEXT J: NEXT I:A = 1: FOR I = 1 TO 11: FOR
     J = 1 TO 11
505  IF K(A) =  ABS (C(J)) THEN  GOSUB 615:H
     $(A) = C$(J) + R$
510  NEXT J
515 A = A + 1: NEXT I
520 H(13) = H(1): FOR I = 1 TO 12:HH = H(I +
     1): IF H(I) > H(I + 1) THEN HH = H(I +
     1) + 360:AB = 1
525 I$ = V$:GG = 0:G = 0
530  VTAB R(I,2): FOR J = 1 TO 11:CC = K(J):
      IF AB = 1 AND CC < 90 THEN CC = CC + 3
     60
535  IF G > 4 THEN GG = 1
540  IF CC > H(I) AND CC < HH THEN  GOSUB 62
     5:G = G + 1
545  NEXT J:AB = 0: NEXT I
555  GOTO 3360
615 R$ = " ": IF C(J) < 0 THEN R$ = "R"
617 C(J) =  ABS (C(J))
620  RETURN
625 Z = K(J): GOSUB 310: IF GG = 1 THEN  PRINT
     I$ TAB( 17) STR$ (I);:GG = 0:I$ = "": GOTO
     635
630  HTAB R(I,1)
632  PRINT  LEFT$ (H$(J),2);: IF  RIGHT$ (H$
     (J),1) = "R" THEN  INVERSE : PRINT "R";
     : GOTO 635
633  PRINT " ";
635  NORMAL : PRINT Z$:H$(J) = "": RETURN
700 SD = (259205536 * T + 2013816) / 3600
710 SD = 17.23 *  SIN ( FN R(C(11))) + 1.27 *
      SIN ( FN R(SD)) - (5025.64 + 1.11 * T)
     * T
720 SD = (SD - 84038.[illegible]) / 3600: RETURN
3360  GOSUB 3460
3370  FOR I = 1 TO 11: INVERSE : VTAB 5: HTAB
     14: PRINT C$(I) SPC( 8)"CENTRO": NORMAL
     :L = L + 1
3390  FOR J = 1 TO 11: IF [illegible] THEN  GOTO
     3440
3392 X$ = CHR$ (J + 48): IF [illegible] THEN X$
     "*"
3393  IF J = 11 THEN X$ = " "
3400  HTAB 17: INVERSE : PRINT [illegible]: NORMAL :
     PRINT " "C$(J)" ": [illegible]
     (J)): IF [illegible]
```

TABELLA DEI PIANETI	
SO	SOLE
LU	LUNA
ME	MERCURIO
VE	VENERE
MA	MARTE
GI	GIOVE
SA	SATURNO
UR	URANO
NE	NETTUNO
PL	PLUTONE
NO	NODO LUNARE NORD

TABELLA DEI SEGNI ASTROLOGICI	
AR	ARIETE
TO	TORO
GE	GEMELLI
CA	CANCRO
LE	LEONE
VE	VERGINE
SC	SCORPIONE
SA	SAGITTARIO
CP	CAPRICORNO
AQ	ACQUARIO
PE	PESCI

Fig. 1.

```
3410  FOR K = 1 TO 8:D =  ABS (AS -  VAL ( RIGHT$
      (A$(K),3))):X$ = "   "
3420  IF D <  VAL ( MID$ (A$(K),4,2)) THEN  PRINT
      "*" LEFT$ (A$(K),3);:K = 9:X$ = "": GOTO
      3435
3430  NEXT K
3435  PRINT X$ RIGHT$ ("   " +  STR$ ( INT (
      S + .5)),3);
3437  GOSUB 3500:Z = ( + .5): GOSUB 310: PRINT
      " " LEFT$ (Z$,4)
3440  NEXT J: GOSUB 3460: NEXT I: PRINT "500
      0000000";: GOSUB 3460: GOTO 3370
3460 ZY = 11: HTAB 14: VTAB 17: INVERSE : PRINT
      "A";: NORMAL : PRINT "SPETTI ";: INVERSE
      : PRINT "N";: NORMAL : PRINT "UOVO"
3470  VTAB 17: HTAB 14: GET G$
3472  IF  ASC (G$) > 48 AND  ASC (G$) < 58 THEN
      I =  VAL (G$) - 1
3473  IF G$ = "*" THEN I = 9
3474  IF G$ = "O" THEN I = 10
3480  IF G$ = "N" THEN  RUN 115
3490  VTAB 7: FOR K = 1 TO 11: HTAB 12: PRINT
      "                        ": NEXT K: RETURN
3500 W = 0:Y =  ABS (C(I) - C(J)): IF Y < =
      180 THEN 3540
3510 W = 180: IF (C(I) + C(J)) / 2 < 180 THEN
      3540
3520 W =  - 180
3540 Y = (C(I) + C(J)) / 2 + W: RETURN
5000  VTAB 8: HTAB (10): GET G$: RETURN
5010  RETURN
7335 MO = 360:A1 =  FN X( SIN (RA) *  TAN (B
      4) *  TAN (OB)): FOR I = 1 TO 12:D =  FN
      U(60 + 30 * I)
7340 A2 = D / 90 - 1:KN = 1: IF D > = 180 THEN
      KN =  - 1:A2 = D / 90 - 3
7345 A3 =  FN R( FN U( FN D(RA) + D + A2 *  FN
      D(A1)))
7350 X =  ATN ( SIN (A3) / ( COS (A3) *  COS
      (OB) - KN *  TAN (B4) *  SIN (OB))): IF
      X < 0 THEN X = X + PI
7355  IF  SIN (A3) < 0 THEN X = X + PI
7360 Z =  FN U( FN D(X) + SD):H(I) = Z: GOSUB
      310:H$(I) = Z$: NEXT I: RETURN
9000 Y = 0:R1 = RA +  FN R(30):FF = 3: GOSUB
      9030:H(5) =  FN U(L + 180):HS$ = "PLACI
      DUS "
9005 R1 = RA +  FN R(60):FF = 1.5: GOSUB 903
      0:H(6) =  FN U(L + 180):FF = 1: GOSUB 9
      030:H(1) = L
9010 R1 = RA +  FN R(120):FF = 1.5:Y = 1: GOSUB
      9030:H(2) = L:R1 = RA +  FN R(150):FF =
      3
9015  GOSUB 9030:H(3) = L:H(4) =  FN U(MC +
      180): FOR I = 1 TO 6:H(I) =  FN U(H(I) +
      SD): NEXT I
9020  FOR I = 1 TO 6:Z =  FN U(H(I) + 180):H
      (I + 6) = Z: GOSUB 310:H$(I + 6) = Z$
9025 Z = H(I): GOSUB 310:H$(I) = Z$: NEXT I:
      RETURN
9030 X =  - 1: IF Y = 1 THEN X = 1
9035  FOR I = 1 TO 10:XX =  FN Y(X *  SIN (R
      1) *  TAN (OB) *  TAN (LA)): IF XX < 0 THEN
      XX = XX + PI
9040 R2 = RA + (XX / FF): IF Y = 1 THEN R2 =
      RA + PI - (XX / FF)
9045 R1 = R2: NEXT I:L =  ATN ( TAN (R1) /  COS
      (OB)): IF L < 0 THEN L = L + PI
9050  IF  SIN (R1) < 0 THEN L = L + PI
9055 L =  FN D(L): RETURN
```

Backgammon

Computer : TRS-80

Modello : mod. 1

Configurazione: 16 K

Autore: Adam Scott

Versione italiana: Pietro Canevarolo

Note: Disponibile su dischetto

Il Backgammon, meglio conosciuto dalle nostre parti come Tric-Trac o Tabla, è un gioco per due persone che si gioca su una scacchiera con due dadi e quindici pedine per ciascuno.

Di antichissima origine orientale, il Backgammon è il gioco nazionale in Grecia e in Egitto. Non molto conosciuto in Italia, sta incontrando ultimamente un grande successo negli Stati Uniti e in Inghilterra dove sono sorti circoli dedicati esclusivamente a questo affascinante gioco.

Regole del gioco

La scacchiera è formata da 24 punte divise in quattro quadranti: un settore interno e uno esterno per ogni giocatore. La posizione iniziale dei pezzi è mostrata in fig. 1. Le pedine vengono mosse da una punta all'altra in base ai numeri che escono dal lancio dei dadi.

Se i due dadi sono diversi, il giocatore può muovere due pedine distinte attribuendo a ognuna il punteggio di un dado, oppure muovere solo una pedina della somma dei due punteggi. I dadi vanno considerati sempre separatamente cosicché anche muovendo una sola pedina non si fa una sola mossa, bensì due.

Ciò significa che se i dadi indicano 2 e 6, uno stesso pezzo può essere mosso prima di sei punte e poi di due oppure prima di due o poi di sei.

Quando una pedina viene spostata su una punta per la prima parte di una mossa si dice che è "atterrata" su di essa.

Se invece i numeri usciti sono uguali, il giocatore deve giocarli quattro volte. Esempio: se esce un doppio due può giocare:

1. una pedina di due punte per quattro volte;
2. due pedine di quattro punte ciascuna;
3. due pedine di due punte e una di quattro;
4. una pedina di sei punte e una di due;
5. quattro pedine di due punte ciascuna.

Se almeno due pedine di uno stesso giocatore si trovano su una punta, esse formano un nastro e nessun pezzo dell'avversario può né fermarvisi né atterrare su quella punta. Una punta bloccata da un nastro non è solo utile per fermare il nemico, ma serve anche come base per le proprie pedine. Se su una punta si trova un solo pezzo questo può venire cacciato (tolto dalla scacchiera) se una pedina avversaria atterra o vi si ferma. Le pedine mangiate vengono messe sulla linea mediana che divide in due parti la scacchiera in senso verticale e devono ricominciare il gioco dall'inizio prima che possa essere effettuata qualsiasi altra mossa, facendo così perdere preziosi turni di gioco.

Quando un giocatore ha riunito tutte le sue pedine nel settore di arrivo (chiamato "casa") può cominciare a toglierle dalla scacchiera in base sempre al lancio dei dadi. Vince chi porta tutti i suoi pezzi fuori dalla scacchiera.

Ci sono tre possibilità di vittoria: normale, se l'avversario ha tolto almeno una pedina; il gammon, che vale il doppio, se l'avversario non è riuscito a togliere nessuna pedina e il backgammon, che vale tre volte la posta, se l'avversario ha pedine fuori oppure nel settore interno dell'avversario.

Per iniziare il gioco si deve tirare un dado ciascuno: inizia chi ha avuto il punteggio più alto.

Strategia elementare

Visto che è meno probabile ottenere punteggi elevati lanciando i dadi di quanto sia possibile ottenerne di bassi, si deve sempre cercare di non accumulare le pedine sulle punte più distanti dal bordo, ma di raccoglierle invece il più possibile vicino all'uscita. Sempre per lo stesso motivo se si riesce ad affiancare più di un nastro, proporzionalmente al numero dei nastri vicini cresce anche la difficoltà per l'avversario a superarli.

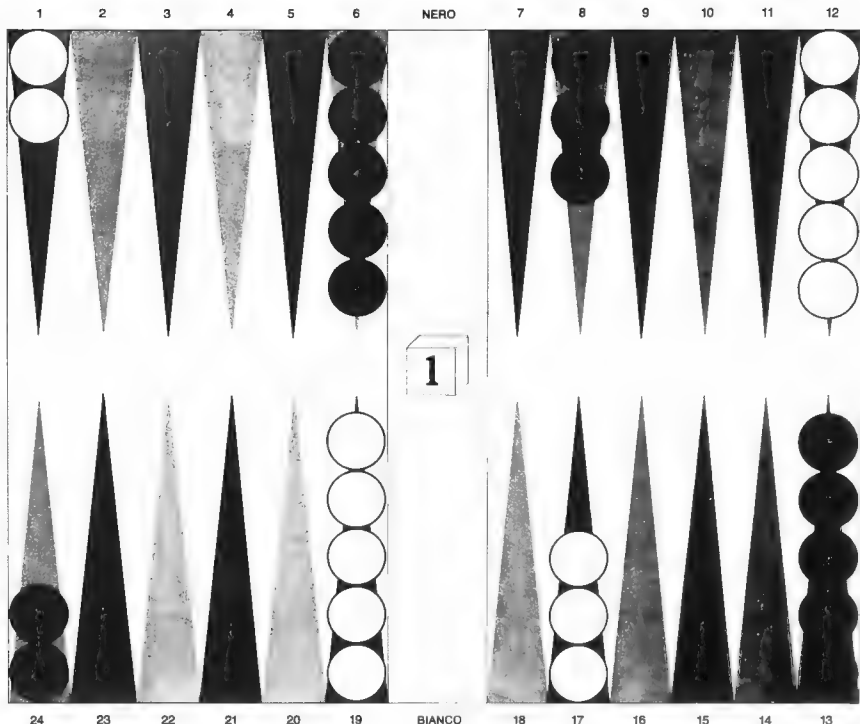
Le pedine isolate sono sempre pericolose, ma, se le pedine dell'avversario sono state tutte superate, allora non ha più importanza se le pedine rimangono isolate a meno che non abbiate intenzione di mangiare qualche pedina del vostro avversario che, rientrando, potrebbe di nuovo minacciare le vostre pedine isolate.

Il programma

Il programma qui descritto provvede a disegnare la scacchiera sullo schermo, tirare i dadi per il computer e per lo sfidante e controllare la validità delle mosse digitate da tastiera. Per eseguire le mosse scrivere il numero della punta di arrivo e quella di partenza, separati da un segno meno (–). Le varie mosse vanno separate tra loro con un punto.

Il calcolatore impiega circa mezzo minuto per ogni mossa, ma se la partita è in una fase critica può "pensare" anche per cinque o dieci minuti di seguito! Ciò è dovuto essenzialmente al fatto che viene utilizzato l'interprete Basic anziché un compilatore, ma il livello del gioco del computer ripaga senz'altro il tempo impiegato per giocare.

Buona fortuna!!!

```
1 REM ********************************************
2 REM *                                          *
3 REM *               BACKGAMMON                 *
4 REM *                                          *
5 REM *         VERSIONE TRS-80 MOD. I           *
6 REM *                                          *
7 REM *           PERSONAL SOFTWARE              *
8 REM *                                          *
9 REM ********************************************
11 CLEAR 100
12 DEFINT A-Z
20 DB=0
25 DIM B(35) :L=0 :M=0 :V=0 :I=0 :J=0 :J1=0 :K=0 :
   J2=0 :J3=0
30 DIM HD(27),DI(4),MV(2,4),BS(2,4),Q7(2,16),DH(4)
32 CLS: PRINT: PRINTTAB(22),"BACKGAMMON": PRINT
33 PRINT"NOTE SUL GIOCO:": PRINT
   "1) SE HAI UNA PEDINA FUORI, DEVI PRIMA FARLA "
   ;"ENTRARE.": PRINT"2) SE NON PUOI MUOVERE BATTI"
   ;" =ENTER=.": PRINT"3) NON VIENI AVVISATO SE "
   ;"DIMENTICHI DI TOGLIERE LE PEDINE."
34 PRINT
35 VH=11 :VT=5 :VP=-7 :VM=5 :VL=-2 :VF=10 :VA=-1 :
   VO=-2
36 PRINT"IO IMPIEGO CIRCA 40 SECONDI PER MOSSA."
   : PRINT"TU PUOI IMPIEGARE QUANTO TEMPO VUOI.":
   PRINT
37 INPUT"HAI MAI GIOCATO CON ME";A$: IF A$="NO"
   GOSUB 20000
40 DATA 2,66,130,194,3,67,131,195,1,65,129,193,0,
   64,128,192,194,130,66,2,195,131,67,3,193,129,
   65,1,192,128,64,0
50 FOR L=1 TO 2: FOR M=1 TO 16: READ Q7(L,M):
   NEXT M,L
60 FOR L=0 TO 27 :B(L)=0 :NEXT :B(24)=-2 :B(1)=2 :
   B(19)=5 :B(6)=-5 :B(12)=5 : B(13)=-5  :B(17)=3
   :B(8)=-3 :LV=1
80 GOSUB 10000
90 GOSUB 11111: IF DI(1)=DI(2) THEN 90 ELSE PRINT
   @896,"IL MIO DADO E'";DI(1);"IL TUO E'";DI(2);
100 IF DI(1)<DI(2) THEN PRINT" PARTI TU."; ELSE
    PRINT"PARTO IO.";
110 FOR L=1 TO 2000 :NEXT
    :IF DI(1)>DI(2) THEN 135 ELSE 120
115 GOSUB 11111
120 GOSUB 1000 :PRINT@896,TAB(63);: PRINT
    :GOSUB 2000
130 GOSUB 11111 :PRINT@960,TAB(63);
135 PRINT@896,TAB(63);: PRINT@896,
    "...UN ATTIMO CHE STO PENSANDO..."
140 PRINT@960,"I MIEI DADI SONO";DI(1);DI(2);
150 GOSUB 5400 :GOSUB 2000
160 GOTO 115
1000 REM
1010 IF NOT POINT(4,19) GOSUB 10000
1020 PRINT@832,TAB(63);: PRINT@832,
     "I TUOI DADI SONO";DI(1);DI(2);". MUOVI";
1025 IF B(26)<>0 AND B(25-DI(1))>1 AND
     B(25-DI(2))>1 PRINT" NON PUOI ENTRARE!";:
     FOR L=1 TO 2000: NEXT: RETURN
1030 A$="" :INPUT A$ :IF LEFT$(A$,1)<>"S" THEN
     1040 ELSE GOSUB 10000 :GOTO 1020
1040 GOSUB 3000 :IF Q<>0 THEN 1095
1060 FOR L=1 TO MS :FOR M=1 TO 2
1070 IF MV(M,L)=0 THEN MV(M,L)=25+M
1080 NEXT M,L
1090 GOSUB 4000
1095 IF Q=0 THEN 1110
1097 PRINT@896,"--ERRORE-- ";
1100 IF Q=2 PRINT"DEVI PRIMA RIENTRARE";
1101 IF Q=1 PRINT"RIPROVA";
1102 IF Q=3 PRINT"POCHE MOSSE";
1103 IF Q=4 PRINT"CONTROLLA I DADI";
1104 IF Q=5 PRINT"TROPPE MOSSE";
1109 PRINT" - HAI MOSSO: ";A$;: PRINT@832,TAB(63);
     : GOTO 1020
1110 IF MS<1 THEN RETURN ELSE FOR I=1 TO MS
1120 IF B(MV(2,I))=1 THEN QF=MV(2,I) :QT=0 :
     GOSUB 9000
1130 QF=MV(1,I) :QT=MV(2,I) :GOSUB 9000
1140 NEXT I :RETURN
2000 REM
2010 IF B(25)=15 THEN LO=LO+1 :A$="O IO!":
     GOTO 2030
2020 IF B(27)<>-15 THEN RETURN ELSE WI=WI+1
     :A$="I TU!"
2030 PRINT@896,TAB(63); :PRINT@896,
     "VINC";A$;" HAI VINTO";WI;"VOLTE E PERSO";
     LO;"PARTITE. ";
2040 INPUT"GIOCHI ANCORA";C$: IF A$="SI" THEN 60
     ELSE CLS: END
3000 REM
3020 IF A$="" THEN MS=0 :Q=0 :RETURN
3030 L=0 : M=0
3035 IF M>3 THEN Q=1 :RETURN
3040 L=L+1 :M=M+1 :GOSUB 3300 :MV(1,M)=Q :L=L+1
3050 GOSUB 3300 :MV(2,M)=Q
3055 IF L>LEN(A$) THEN 3100
3060 IF MID$(A$,L,1)<>"-" THEN 3090
3070 L=L+1 :GOSUB 3300 :IF M+Q-1>DN THEN
     Q=5 :RETURN
3080 N=M :FOR M=N+1 TO N+Q-1 :MV(1,M)=MV(1,N)
     :MV(2,M)=MV(2,N): NEXT :M=N+Q-1
3090 IF L<LEN(A$) THEN IF MID$(A$,L,1)="."
     THEN 3035 ELSE Q=1 :RETURN
3100 Q=0 :MS=M :RETURN
3300 REM
3310 B$=MID$(A$,L,1) :C$=MID$(A$,L+1,1)
3320 IF C$>="0" AND C$<="9" THEN L=L+1 :B$=B$+C$
3330 Q=VAL(B$) :L=L+1 :RETURN
4000 REM
4010 FOR L=0 TO 27 :HD(L)=B(L) :NEXT :Q=1
4015 IF MS=0 THEN 4110
4020 FOR L=1 TO MS :FOR M=1 TO 2 : N=MV(M,L)
4030 IF N<1 OR N>27 OR N=25 THEN 4200
4040 NEXT M,L
4050 FOR L=1 TO MS :QT=MV(2,L) :QF=MV(1,L)
4060 IF B(QT)>1 OR B(QF)>-1 THEN 4200
4062 IF B(26)<>0 AND (QF<>26 OR QT<19) THEN
     Q=2 :GOTO 4200
4070 B(QF)=B(QF)+1 :B(QT)=B(QT)-1 :IF B(QT)=0
     THEN B(QT)=-1
4090 IF QT=27 THEN N=B(26) :FOR M=7 TO 24 :
     N=N+(B(M)<0) :NEXT: IF N<>0 THEN 4200
4100 NEXT L
4110 GOSUB 6500
4120 FOR L=0 TO 27 :B(L)=HD(L) :NEXT :RETURN
4200 GOTO 4120
5000 REM
5010 FOR L=0 TO 27 :HD(L)=B(L) :NEXT L: ZZ=Z
```

```
5020 V=0 :FOR L=1 TO DN :ON L GOTO 5025,
     5030, 5035, 5040
5025 M=J :M1=J+D :GOTO 5044
5030 M=J1 :M1=J1+D1 :GOTO 5044
5035 M=J2 :M1=J2+D :GOTO 5044
5040 M=J3 :M1=J3+D
5041 REM
5044 IF Z=1 AND M1>25 AND M1<999
     FOR M9=18 TO M-1 :IF B(M9)<1 THEN NEXT
     M9:M1=25
5045 IF M1>24+Z IF M1<999 THEN V=-10000
     :GOTO 5070
5046 IF M>24 OR M1>24+Z THEN V=V-100 :GOTO 5060
5048 B(M)=B(M)-1
5050 IF B(M)>-1 THEN B(M1)=B(M1)+1 :IF B(M1)=0
     THEN B(M1)=1 :V=V+VT :B(26)=B(26)-1 :IF
     M1<11 THEN V=V+VT
5055 IF B(M)<0 IF B(25)>14 THEN V=10000 :
     GOTO 5070 :ELSE V=-10000 :GOTO 5070
5057 IF Z=1 AND M1=25 THEN V=V+VF
5058 IF Z=0 FOR LL=0 TO 18 :IF B(LL)<1
     NEXT LL :Z=1
5060 NEXT L :GOSUB 8000
5070 FOR L=0 TO 27: B(L)=HD(L) :NEXT L :
     Z=ZZ :RETURN
```

```
5200 REM
5201 IF DB=1 PRINT@832,"DAL 'BAR'":
5210 IF B(0)=0 OR DN<1 THEN RETURN
5220 FOR I=1 TO DN : J=DI(I) :IF B(J)<>-1 THEN
     5240 ELSE QF=J :QT=26 :GOSUB 9000: QF=0 :
     QT=J :GOSUB 9000
5230 DI(I)=DI(DN) :DN=DN-1 :GOTO 5210
5240 NEXT I :FOR I=1 TO DN: J=DI(I) :IF B(J)=1
     THEN QF=0 :QT=J :GOSUB 9000 :GOTO 5230
5250 NEXT I :FOR I=1 TO DN: J=DI(I) :IF B(J)>-1
     THEN QF=0 :QT=J :GOSUB 9000 :GOTO 5230
5260 NEXT I :PRINT@CR," NON POSSO MUOVERE!";:
     RETURN
5400 REM
5410 PRINT". MUOVO: "; :CR=960+POS(0) :Z=0
5411 IF DB=1 PRINT@832,TAB(63);
5420 L=0 :FOR I=0 TO 18 :L=L+(B(I)>0) :NEXT I
5430 IF L=0 THEN Z=1 :LV=2
5440 GOSUB 5200
5450 IF DN=0 OR B(0)<>0 THEN 5470
5452 IF Z=0 THEN 5460
5455 IF B(26)=0 FOR L=1 TO DN:
     FOR M=25-DI(L) TO 24 :IF B(M) -1 NEXT M:
     IF B(25-DI(L))<1 THEN NEXT L ELSE
     QF=25-DI(L) :QT=25 :GOSUB 9000 :DI(L)=0 :
```

```
     NEXT L
5456 M=0 :FOR L=1 TO DN :IF DI(L)<>0 THEN
     M=M+1 :DH(M)=DI(L)
5457 NEXT L :IF DN<>M :FOR L=1 TO DN :DI(L)=
     DH(L) :NEXT L :DN=M
5458 IF DN<1 THEN 5450
5460 IF DN<3 THEN GOSUB 7000 ELSE GOSUB 6000
5470 PRINT@896,TAB(63); :RETURN
5700 REM
5705 IF DB=1 RETURN
5710 IF FG=0 THEN FG=1
5720 FG=-FG :PT=896+17
5730 IF FG<0 THEN PRINT@PT,CHR$(30);"PENSO...";
     ELSE PRINT@PT,CHR$(30);"STO PENSANDO...";
5740 RETURN
6000 REM
6001 IF DB=1 PRINT@832,"MOSSA DOPPIA.";
6010 MX=-9999 : D=DI(1) :D1=D :E=24 :NM=0
     :J=999 :J1=999 :J2=999 :J3=999
6012 BG=2*VL :IF LV=2 THEN 6019
6014 FOR L=1 TO 24 :IF B(L)<>-1 THEN NEXT L
     ELSE: K=L: FOR M=1 TO DN/2 :K=K-D :IF K<1
     THEN NEXT L ELSE IF B(K)<-1 THEN NEXT L ELSE
     IF B(K)>1 THEN BG=1  ELSE NEXT M,L
6019 IF Z=1 THEN BG=VF*DN/2+VM
6020 FOR J=1 TO E :K=B(J)
6030 IF K>0 THEN K=B(J+D) :IF K>-2 THEN 6050
6035 IF Z=1 IF J+D>25 THEN 6050
6040 NEXT J :J=999 :IF MX>-50 THEN 6150 ELSE 6140
6050 FOR J1=J TO E :K=B(J1) :IF J1=J+D THEN K=1
6055 IF J1=J IF K<2 THEN K=0
6060 IF K>0 THEN K=B(J1+D) :IF K>-2 THEN 6080
6065 IF Z=1 IF J1+D1>25 THEN 6080
6070 NEXT J1: J1=999 :IF MX>-50 THEN 6040 ELSE 6140
6080 GOSUB 5700 :FOR J2=J1 TO E :K=B(J2)
     :IF J2=J1+D OR J2=J+D THEN K=1
6085 IF J2=J1 IF J1=J IF K<3 THEN K=0
6090 IF K>0 THEN K=B(J2+D) :IF K>-2 THEN 6110
6095 IF Z=1 IF J2+D>25 THEN 6110
6100 NEXT J2 :J2=999 :IF MX>-50 THEN 6070
     ELSE 6140
6110 GOSUB 5700: IF DN<4 THEN 6140 ELSE FOR J3=J2
     TO E :K=B(J3) : IF J3=J1+D OR J3=J2+D OR
     J3=J+D THEN K=1
6120 IF K>0 THEN K=B(J3+D) :IF K>-2 THEN 6140
6125 IF Z=1 IF J3+D>25 THEN 6140
6130 NEXT J3 :J3=999 :IF MX>-50 THEN 6100
6140 GOSUB 5000 :IF V>MX THEN MX=V: NM=1
     :BS(1,1)=J :BS(2,1)=J+D :BS(1,2)=J1 :
     BS(2,2)=J1+D:BS(1,3)=J2 : BS(2,3)=J2+D :
     BS(1,4)=J3 :BS(2,4)=J3+D:IF DB=1THEN
     PRINT"*";V;J;J1;J2;J3;
6141 IF MX>=BG THEN 6160
6145 IF J3<>999 THEN 6130 ELSE IF J2<>999
     THEN 6100 ELSE IF J1<>999 THEN 6070
     ELSE IF J<>999 THEN 6040
6150 REM
6160 GOTO 7120
6500 REM
6510 FOR L=1 TO 4 :DH(L)=DI(L) :NEXT L
6515 IF MS=0 THEN 6570
6520 FOR L=1 TO MS
6530 QF=MV(1,L) :IF QF=26 LET QF=25
6540 QT=MV(2,L) :IF QT=27 LET QT=0
6550 FOR M=1 TO DN :IF QF-QT=DH(M) THEN DH(M)=0 :
     ELSE NEXT M
6560 NEXT L
6570 FOR L=1 TO DN
6580 IF DH(L)=0 THEN 6700
6590 REM
6600 D=DH(L) :IF B(26)<>0 IF B(25-D)>0 THEN 6700
     ELSE Q=2 :GOTO 6710
6605 Q=3 :IF MS=DN THEN Q=4
6610 FOR M=1 TO 24
6620 IF B(M)>-1 THEN 6650
6630 N=M-D :IF N<1 THEN 6650
6640 IF B(N)<1 THEN 6710
6650 NEXT M
6700 NEXT L :Q=0
6710 RETURN
7000 REM
7001 IF DB=1 PRINT@832,"REGOLAMENTARE.";
7010 MX=-9999 : NM=0
7020 FOR I=1 TO DN :D=DI(I) :D1=DI(3-I) :J1=999
7030 FOR J=1 TO 25-D+Z*(D-1) :K=B(J)
7040 IF K>0 THEN K=B(J+D) :IF K>-2 THEN 7060
7045 IF Z=1 IF J+D>25 THEN 7060
7050 NEXT J :J=999 :IF MX>-50 THEN 7110 ELSE 7100
7060 IF DN=1 THEN 7100
7065 K=J :IF K<24 IF I=2 OR B(K)<2 LET K=K+1
7070 GOSUB 5700 :FOR J1=K TO 24 :K=B(J1)
     :IF J1=J+D THEN K=1
7080 IF K>0 THEN K=B(J1+D1) :IF K>-2 THEN 7100
7085 IF Z=1 IF J1+D1>25 THEN 7100
7090 NEXT J1 :J1=999 :IF MX>-50 THEN 7050
7100 GOSUB 5000 :IF V>MX THEN MX=V: NM=1
     :BS(1,1)=J :BS(2,1)=J+D
     :BS(1,2)=J1 :BS(2,2)=J1+D1
     :IF DB=1 PRINT"*";MX;J;J+D;J1;J1+D1;
7105 IF J1<>999 THEN 7090 ELSE IF J<>999 THEN 7050
7110 NEXT I
7120 IF NM=0 PRINT@CR,"NON POSSO MUOVERE!";:
     RETURN
7130 NM=0 :FOR L=1 TO DN :FOR M=1 TO 2
7133 IF BS(M,L)>25 AND BS(M,L)<>999 THEN
     BS(M,L)=25
7135 IF BS(M,L)<0 OR BS(M,L)>=999 THEN 7160
     ELSE NEXT M
7140 IF B(BS(2,L))=-1 THEN QF=BS(2,L) :QT=26 :
     GOSUB 9000
7150 QF=BS(1,L) :QT=BS(2,L) :GOSUB 9000 :NM=1
7160 NEXT L :IF NM=0 THEN 7120 ELSE RETURN
8000 REM
8010 IF Z=1 THEN 8040
8020 FOR L=0 TO 18 :IF B(L)>0 THEN 8030 ELSE
     NEXT L: V=V+VM
8030 FOR L=1 TO 6 :IF B(L)>0 THEN V=V+B(L)*VL
8035 NEXT L
8040 M=26 :IF B(26)<>0 THEN 8070
8050 M=0 :FOR L=24 TO 0 STEP -1 :IF M=0 IF
     B(L)<0 LET M=L
8060 IF M=0 OR B(L)<=0 THEN NEXT :V=V+VH :GOTO
     8100
8065 REM
8070 FOR L=1 TO 24
8075 IF B(L)<>1 THEN 8090
8080 IF L<7 THEN V=V+VP-VL :GOTO 8090
8085 IF L>M THEN V=V+VA ELSE V=V+VP
8087 IF L>18 THEN V=V-VO*B(26)
8090 NEXT L
8095 IF B(26)=0 THEN 8100 ELSE M=0 :FOR
     L=19 TO 24:IF B(L)=1 THEN M=0 ELSE IF
     B(L)>1 THEN M=M+1 :NEXT L :ELSE NEXT L
8096 IF M<>6 THEN V=V+(M*3-1)/2 ELSE V=V+35
8100 RETURN
9000 REM
9010 S=SGN(B(QF)) :IF S=0 RETURN
9020 Q4$="O" :IF S=-1 THEN Q4$="X"
9030 Q=ABS(B(QF)) :B(QF)=B(QF)-S :Q2=QF
9040 GOSUB 9100 :PRINT@Q1," ";
9050 B(QT)=B(QT)+S :Q=ABS(B(QT)) :Q2=QT
9060 GOSUB 9100 :PRINT@Q1,Q4$;
9065 B$=MID$("##",1,2+(QF<10))
     :C$=MID$("##",1,2+((QT<10)OR(QT=25)))
9070 IF S=1 PRINT@CR,""; :PRINTUSING B$;QF; :PRINT
     "-";: PRINTUSING C$;-QT*(QT<25); :PRINT".";:
     CR=960+POS(0)
9080 RETURN
9100 REM
9110 Q3=1 :IF Q2>0 AND Q2<25 THEN Q3=CINT(Q2/13)+1
9120 IF Q2>25 THEN Q3=2
9125 Q3=3-Q3
9130 IF Q2=0 OR Q2=26 THEN Q4=155+320*(2-Q3) ELSE
     IF Q2=25 OR Q2=27 THEN Q4=187+320*(2-Q3) ELSE
     IF Q2>18 THEN Q4=128+33+(Q2-19)*4 ELSE
     IF Q2>12 THEN Q4=128+(Q2-13)*4 ELSE
     IF Q2>6 THEN Q4=448+(12-Q2)*4 ELSE
     Q4=448+33+(6-Q2)*4
9140 Q1=Q7(Q3,Q)+Q4 :RETURN
10000 REM
10010 CLS :PRINT@768,; :FOR L=12 TO 1 STEP -1 :
      PRINTUSING"  ##";L;:IF L=7 THEN PRINT
      "   BAR   "; ELSE IF L=1 PRINT"  FUORI";
10020 NEXT :PRINT@0,;: FOR L=13 TO 24 :PRINTUSING
      "  ##";L;: IF L=18 THEN PRINT"   BAR   ";
      ELSE IF L=24 PRINT"  FUORI";
```

```
10025 NEXT
10030 PRINT@384,; :FOR L=0 TO 56 STEP 2 :PRINT
      CHR$(140);CHR$(032); :NEXT :PRINT@384," ";
10040 FOR L=65 TO 118 STEP 4 :FOR M=L TO
      L+640 STEP 640
10042 IF L>86 AND L<97 THEN 10049
10043 N=M :IF L>90 LET N=N+1
10045 K=184 :K1=180 :IF M=L THEN K=139 :K1=135
10048 PRINT@N,CHR$(K);CHR$(191);CHR$(K1);
10049 NEXT M,L
10050 FOR L=49 TO 65 STEP 16 :FOR M=0 TO 37 :
      SET(L,M) :NEXT M,L
10060 REM
10070 FOR Q2=0 TO 27 :IF B(Q2)=0 THEN 10110
10080 FOR Q=1 TO ABS(B(Q2)) :GOSUB 9100
10090 IF B(Q2)<0 THEN PRINT@Q1,"X"; ELSE
      PRINT@Q1,"O";
10100 NEXT
10110 NEXT
10120 RETURN
11111 FOR RR=1 TO RND(100) :DI(1)=RND(6) :NEXT
      :RANDOM
11121 FOR RR=1 TO RND(100) :DI(2)=RND(6) :NEXT
      :RANDOM
11130 IF DI(1)=DI(2) THEN DN=4 :DI(3)=DI(1)
      :DI(4)=DI(1): RETURN
11140 DN=2 :RETURN
20000 REM ISTRUZIONI
20010 CLS: PRINT"LE REGOLE SONO QUELLE ";
      "INTERNAZIONALI."
20020 PRINT"SCRIVI LE MOSSE DA-A-VOLTE E ";
      "SEPARA OGNI MOSSA CON UN PUNTO."
20025 PRINT
20030 PRINT"ESEMPIO: PER MUOVERE DUE VOLTE ";
      "DA 2 A 1."
20040 PRINT"SCRIVI:";TAB(25);"2-1-2"
20045 PRINT
20050 PRINT"PER MUOVERE DA 3 A 2 E DA 13 A 8, "
      : PRINT"SCRIVI:";: PRINTTAB(25);"3-2.13-8"
20060 PRINT
20070 PRINT"PER MUOVERE DA 7 A 6, DA 5 A 4 E ";
      "TOGLIERE DUE PEDINE DALL'1": PRINT
      "SCRIVI:";: PRINTTAB(25)"7-6.5-4.1-0-2"
20080 PRINT
20090 PRINT"NOTA CHE SIA CHE IL 'BAR' CHE ";
      "IL FUORI SI INDICANO CON 0."
20100 PRINT"SCRIVI SCACCHIERA SE NON CAPISCI ";
      "A CHE PUNTO E' IL GIOCO."
20400 INPUT"SE SEI PRONTO PREMI =ENTER=";A$:
      RETURN
```

Collisione

Computer: Apple II

Modello: Europlus

Configurazione: 32 K

Autore: A. Giovannetti

Note: Disponibile su dischetto

Questo è uno dei primi giochi elettronici apparsi nei nostri bar qualche anno fa.

La base del gioco è un tabellone composto di cinque cornici concentriche, ognuna ricoperta di mine che devono essere eliminate dal giocatore. Il vostro veicolo è un razzo che segue le piste in senso antiorario, mentre il computer sposta il suo in senso orario cercando di scontrarsi con il vostro.

Per fortuna ci sono quattro punti sul percorso, nei quali potete cambiare corsia ruotando il potenziometro della paletta n. 1. Potete anche raddoppiare la vostra velocità premendo il pulsante sulla paletta.

Ogni mina che riuscite a togliere dal percorso ha il valore di un punto. Ogni tanto il computer lascia sul suo percorso un altro tipo di mine (più grandi) che valgono invece cinque punti. Se riuscite a ripulire un intero tabellone, automaticamente ne ricomparirà uno nuovo.

Il programma usa una tabella di tre tipi di caratteri speciali: il primo per disegnare le mine, il secondo per le mine da cinque punti e il terzo per disegnare il vostro razzo. Esso usa pure le istruzioni PEEK e POKE per memorizzare il tabellone, invece di una matrice che avrebbe occupato più memoria e avrebbe reso l'elaborazione notevolmente più lenta.

Notate che in tutto il programma viene usata una sola istruzione DATA.

Variabili principali

A, A$, B sono usate con l'istruzione POKE per memorizzare i caratteri speciali per le mine e il razzo.

C conta le mine rimaste.

DC, DD, DY contengono un identificatore della di-

rezione del movimento dei razzi.
HS è il punteggio massimo ottenuto fino all'ultimo turno.
L è un puntatore per le tabelle in memoria.
I è un contatore di ciclo.
LC, LD, LY indicano in quale cornice si trovano i razzi (1 per la più interna, 5 per la più esterna).
R è un numero casuale che determina la posizione delle mine.
RK contiene il numero delle cornici svuotate.
S è la prima locazione di memoria che contiene il tabellone.
SC contiene il punteggio attuale.
SW stabilisce quale dei due razzi del computer deve muoversi (solo dopo che sono stati completati due tabelloni).
T conta i passi del razzo del giocatore (se il pulsante è premuto, vale il doppio).
X, Y coordinate del tabellone.
XC, YC, XD, YD, XY, YY coordinate dei tre razzi (due del computer e uno del giocatore).
Z usata per immagazzinare i risultati parziali dei calcoli.

```
 1 REM *******************************************
 2 REM *                                         *
 3 REM *              COLLISIONE                 *
 4 REM *          VERSIONE APPLE II              *
 5 REM *                                         *
 6 REM *          PERSONAL SOFTWARE              *
 7 REM *                                         *
 8 REM *******************************************
48  HOME
50  INPUT "VUOI LE ISTRUZIONI? ";ZZ$
53  IF  LEFT$ (ZZ$,1) = "S" THEN  GOSUB 5000
55  GOTO 1000
60  IF  PEEK ( - 16287) > 127 THEN T = 2
65  HCOLOR= 0: ROT= DY * 16: DRAW 3 AT 6 + 1
     2 * XY,6 + 12 * YY
70  IF  ABS (XY - 11) > 1 AND  ABS (YY - 6)
     > 1 THEN 110
75  ON DY GOTO 80,85,90,95
80  IF XY = 12 THEN 150
82 Z =  INT ( PDL (0) / 86) - 1:LY = LY + Z:
     YY = YY + Z: IF LY < 1 OR LY > 5 THEN L
     Y = LY - Z:YY = YY - Z
84  GOTO 150
85  IF YY = 5 THEN 150
87 Z =  INT ( PDL (0) / 86) - 1:LY = LY + Z:
     XY = XY - Z: IF LY < 1 OR LY > 5 THEN L
     Y = LY - Z:XY = XY + Z
89  GOTO 150
90  IF XY = 10 THEN 150
92 Z =  INT ( PDL (0) / 86) - 1:LY = LY + Z:
     YY = YY - Z: IF LY < 1 OR LY > 5 THEN L
     Y = LY - Z:YY = YY + Z
94  GOTO 150
95  IF YY = 7 THEN 150
97 Z =  INT ( PDL (0) / 86) - 1:LY = LY + Z:
     XY = XY + Z; IF LY < 1 OR LY > 5 THEN L
     Y = LY - Z:XY = XY - Z
99  GOTO 150
110  IF  ABS (XY - 11) < LY + 6 OR  ABS (YY -
     6) < LY + 1 THEN 150
120 DY = DY - 1: IF DY = 0 THEN DY = 4
150  ON DY GOTO 160,170,180,190
160 XY = XY + 1: GOTO 200
170 YY = YY + 1: GOTO 200
180 XY = XY - 1: GOTO 200
190 YY = YY - 1
200  IF XY = XC AND YY = YC THEN 1013
205  IF RK > 2 THEN  IF XY = XD AND YY = YD THEN
     1013
210 Z = XY * 13 + YY + S: IF  PEEK (Z) < 1 THEN
     240
220  ROT= 0: DRAW  PEEK (Z) AT XY * 12 + 6,Y
     Y * 12 + 6:SC = SC + ( PEEK (Z) - 1) *
     4 + 1:C = C - 1: POKE Z,0: VTAB 22: HTAB
     1: PRINT SC
230  IF C = 0 THEN 1015
240  IF T = 2 THEN T = 1: GOTO 110
250  HCOLOR= 7: ROT= DY * 16: DRAW 3 AT XY *
     12 + 6,YY * 12 + 6
260  IF RK > 2 THEN T = 2:SW = 1 - SW: ON SW
      + 1 GOTO 300,500
300  HCOLOR= 0: ROT= DC * 16: DRAW 3 AT XC *
     12 + 6,YC * 12 + 6
305 Z =  PEEK (XC * 13 + YC + S): IF Z THEN
      HCOLOR= Z: ROT= 0: DRAW Z AT XC * 12 +
     6,YC * 12 + 6
310  IF XC < > 11 AND YC < > 6 THEN 360
315 R = 0: IF  RND (1) < .05 THEN R = 1
320 Z =  SGN (LY - LC):LC = LC + Z
330  ON DC GOTO 335,340,345,350
335 YC = YC - Z: GOTO 400
340 XC = XC + Z: GOTO 400
345 YC = YC + Z: GOTO 400
350 XC = XC - Z: GOTO 400
360  IF  ABS (XC - 11) < LC + 6 OR  ABS (YC -
     6) < LC + 1 THEN 400
370 DC = DC + 1: IF DC = 5 THEN DC = 1
400  ON DC GOTO 410,420,430,440
410 XC = XC + 1: GOTO 450
420 YC = YC + 1: GOTO 450
430 XC = XC - 1: GOTO 450
440 YC = YC - 1
450  IF XC = XY AND YC = YY THEN 1013
455  IF  NOT R THEN 470
457 Z = XC * 13 + YC + S: IF  PEEK (Z) = 0 THEN
     C = C + 1
460  POKE Z,2
470  IF T = 2 THEN T = 1: GOTO 700
480  HCOLOR= 5: ROT= DC * 16: DRAW 3 AT XC *
     12 + 6,YC * 12 + 6
490  GOTO 60
500  HCOLOR= 0: ROT= DD * 16: DRAW 3 AT XD *
     12 + 6,YD * 12 + 6
505 Z =  PEEK (XD * 13 + YD + S): IF Z THEN
      HCOLOR= Z: ROT= 0: DRAW Z AT XD * 12 +
     6,YD * 12 + 6
510  IF XD < > 11 AND YD < > 6 THEN 560
520 Z =  SGN (LY - LD):LD = LD + Z
530  ON DD GOTO 535,540,545,550
535 YD = YD - Z: GOTO 600
540 XD = XD + Z: GOTO 600
545 YD = YD + Z: GOTO 600
550 XD = XD - Z: GOTO 600
560  IF  ABS (XD - 11) < LD + 6 OR  ABS (YD -
     6) < LD + 1 THEN 600
570 DD = DD + 1: IF DD = 5 THEN DD = 1
600  ON DD GOTO 610,620,630,640
610 XD = XD + 1: GOTO 650
620 YD = YD + 1: GOTO 650
630 XD = XD - 1: GOTO 650
640 YD = YD - 1
650  IF XD = XY AND YD = YY THEN 1013
652  IF XD = XY AND YD = YY THEN 1013
655  IF T = 2 THEN T = 1: GOTO 500
660  HCOLOR= 5: ROT= DD * 16: DRAW 3 AT XD *
     12 + 6,YD * 12 + 6
670  GOTO 60
1000  GOSUB 1300
1010  POKE 232,0: POKE 233,30
1011 HS = 0:SC = 0:SW = 0
1013  IF SC > HS THEN HS = SC
1014 SC = 0:RK = 0
1015 C = 200:RK = RK + 1
```

```
1020  HGR : HCOLOR= 6
1030  FOR Y = 0 TO 60 STEP 12
1040  HPLOT Y,Y TO 276 - Y,Y TO 276 - Y,156 -
    Y TO Y,156 - Y TO Y,Y
1050  NEXT
1060  HCOLOR= 0
1070  FOR Y = 12 TO 48 STEP 12
1080  HPLOT Y,65 TO Y,91
1090  HPLOT 276 - Y,65 TO 276 - Y,91
1100  HPLOT 125,Y TO 151,Y
1110  HPLOT 125,156 - Y TO 151,156 - Y
1120  NEXT
1121 S = 7716
1122  FOR L = S TO S + 298: POKE L,0: NEXT
1125  HCOLOR= 1: ROT= 0: SCALE= 1
1130  FOR Y = 0 TO 4
1140  FOR X = 0 TO 9
1150  DRAW 1 AT 6 + 12 * X,6 + 12 * Y: POKE
    13 * X + Y + S,1
1160  DRAW 1 AT 270 - 12 * X,6 + 12 * Y: POKE
    13 * (22 - X)*+ Y + S,1
1170  DRAW 1 AT 6 + 12 * X,150 - 12 * Y: POKE
    13 * X + (12 - Y) + S,1
1180  DRAW 1 AT 270 - 12 * X,150 - 12 * Y: POKE
    13 * (22 - X) + (12 - Y) + S,1
1190  NEXT X: NEXT Y
1220 XC = 10:YC = 12:DC = 3:LC = 5
1230 XY = 12:YY = 12:DY = 1:LY = 5
1240  ROT= DY * 16: HCOLOR= 7: DRAW 3 AT XY *
    12 + 6,YY * 12 + 6
1250  ROT= DC * 16: HCOLOR= 5: DRAW 3 AT XC *
    12 + 6,YC * 12 + 6
1255  IF RK > 2 THEN XD = 10:YD = 0:DD = 1:L
    D = 5: ROT= DD * 16: DRAW 3 AT XD * 12 +
    6,YD * 12 + 6
1260  HOME : VTAB 21: PRINT "PUNTI","RECORD
      ": PRINT SC,HS
1270  FOR I = 1 TO 1000: NEXT : GOTO 60
1300 L = 7680
1310 A$ = "030008000B0012003E2C003E2E2E2C3C3
    C0037373F3E362E2C252D352D2E243C3C272700"
1320  FOR I = 1 TO 72 STEP 2
1330 A =  ASC ( MID$ (A$,I,1)) - 48: IF A >
    9 THEN A = A - 7
1340 B =  ASC ( MID$ (A$,I + 1,1)) - 48: IF
    B > 9 THEN B = B - 7
1350  POKE L,A * 16 + B:L = L + 1: NEXT I: RETURN

4999  END
5000  HOME : VTAB (2): HTAB (15): INVERSE : PRINT
    "COLLISIONE": NORMAL
5050  PRINT : PRINT : PRINT
5100  PRINT "DEVI GUIDARE UN RAZZO RACCOGLI-
    MINE": PRINT "ATTRAVERSO CINQUE CORNICI
     CONTRICHE"
5200  PRINT "RACCOGLIENDO PIU' MINE POSSIBIL
    I": PRINT "SENZA PERO' SCONTRARTI CON I
    L RAZZO "
5300  PRINT "DEL COMPUTER."
5350  PRINT "IL TUO RAZZO GIRA IN SENSO ANTI
    ORARIO.": PRINT "QUELLO DEL COMPUTER IN
     SENSO ORARIO."
5400  PRINT "USANDO LA PALETTA N. 1 PUOI CAM
    BIARE": PRINT "CORSIA, EVITANDO COSI' L
    E COLLISIONI."
5450  PRINT "CON IL PULSANTE DELLA PALETTA P
    UOI": PRINT "RADDOPPIARE LA TUA VELOCIT
    A'.": PRINT "IL COMPUTER, PASSANDO NELL
    E CORSIE "
5500  PRINT "VUOTE, DEPONE DUE TIPI DI MINE-
    ": PRINT "LE PIU' PICCOLE VALGONO UN PU
    NTO": PRINT "QUELLE PIU' GRANDI NE VALG
    ONO CINQUE."
5550  PRINT : INPUT "SE SEI PRONTO PREMI RET
    URN ";ZZ$: HOME
9999  RETURN
```

Rally

Computer: Acorn	**Autore:** Pietro Canevarolo
Modello: Atom	
Configurazione: espanso	

Immaginatevi alla guida di una veloce macchina da corsa in una strada piena di curve. Le curve sono imprevedibili e dovete fare molta attenzione nella guida o rischiate di andare fuori strada. Quanta strada potete fare in un giorno? Per quanto tempo riuscirete a sostenere una guida veloce intorno al circuito? Con questo programma avrete brivido a volontà e senza uscire di casa.

La difficoltà del gioco è sotto il vostro controllo: larghezza della strada e velocità dell'auto vi vengono richieste all'inizio del gioco.

Uso

Il programma si presenta con una breve introduzione grafica e con alcune spiegazioni su come guidare l'auto. In seguito vi chiede larghezza della strada e velocità dell'auto. La larghezza può variare da 4 caratteri a 15.

Più stretta è la strada, più difficile è il gioco. La velocità varia da 1 a 100, ovviamente è più difficile guidare un'auto a cento chilometri all'ora che farlo a passo d'uomo. Un conteggio alla rovescia precede la partenza della gara: dovrete stare pronti per evitare di uscire di strada già alle prime curve.

La guida della macchina viene effettuata con i tasti SHIFT e REPT nell'angolo a destra della tastiera. Premendo il tasto SHIFT l'auto si dirigerà a sinistra, mentre con il tasto REPT succederà il contrario. Se non premerete nessun tasto l'auto proseguirà diritta nella sua corsa.

La corsa continua finché l'auto non va fuori strada, dopo di che vi verrà mostrato uno schema riassuntivo dei chilometri percorsi e dei giorni di corsa impiegati.

Dopo ogni collisione potete scegliere se continuare la corsa, premendo il tasto C, oppure ricominciare la gara, con il tasto R, o, come ultima possibilità, abbandonare la corsa con il tasto A.

Sia premendo A che R vi viene mostrata la media chilometrica giornaliera che siete riusciti a mantenere.

Ci sono diversi modi di giocare: potete vedere quanta strada riuscite a fare in un certo numero di giorni, oppure cercare di percorrere un certo numero di chilometri, poniamo 1000, nel minor numero di giorni possibile.

Quando vi sarete impratichiti nel gioco, provate cambiando la larghezza della strada o la velocità della macchina: il programma continuerà a mettere alla prova la vostra abilità nella guida e voi continuerete a divertirvi.

```
 10 DIM AA(2)
 15 ?#80=0
 20 A=20
175 GOSUB 1400
180 GOSUB 800
183 FOR I=1 TO 9000; NEXT
184 PRINT$12
185 PRINT''"USA IL TASTO 'SHIFT' PER"'"GIRARE A SINISTRA"'
190 PRINT"USA IL TASTO 'REPT' PER"'"GIRARE A DESTRA"'
195 PRINT"SE NON PREMI NESSUN TASTO"'"L'AUTO PROSEGUE DIRITTA"
197 PRINT'
200 T=0; N=0
210 INPUT'"LARGHEZZA DELLA STRADA (4-15)"W
220 IF W<4 OR W>15 GOTO 210
230 INPUT'"VELOCITA' (1-100)"V
```

```
 300 N=N+1
 310 C=(D+S)/2; H=0
 318 S=13; D=S+W; C=(D+S)/2
 320 FOR J=1 TO 16; GOSUB 600; PRINT'; NEXT
 330 GOSUB 700
 340bREM
 345 IF H=1 H=2; GOTO 375
 350 H=H+1; Q=RND%100
 355 IF Q>50 IF D<32 PRINT'; GOSUB 640; GOTO a
 360 IF Q<50 IF S>1 PRINT'; GOSUB 620; GOTO a
 370 PRINT'
 375 GOSUB 600
 400aIF ?#B001&#80=0 C=C-1
 410 IF ?#B002&#40=0 C=C+1
 415 FOR I=1000 TO V*10 STEP-1; NEXT
 420 IF C>S IF C<D+1 GOSUB 1000; GOTO b
 440 GOSUB 1000; FOR K=1 TO 50; NEXT
 445 @=3
 450 M=H*5; T=T+M; PRINT'"HAI FATTO "M" CHILOMETRI"'
 460 PRINT"PER UN TOTALE DI "T" CHILOMETRI IN "N" GIORN"
 465 IF N=1 PRINT"O"'
 466 IF N<>1 PRINT"I"'
 467 PRINT"BATTI 'C' PER CONTINUARE"'
 470 PRINT"       'R' PER RIPARTIRE"'
 472 PRINT"       'A' PER ABBANDONARE"'
 500 LINK AA0; IF ?#80=CH"C" PRINT$12; GOTO 300
 510 IF ?#80<>CH"R" IF ?#80<>CH"A" GOTO 500
 520 PRINT"MEDIA DI "T/N" CHILOMETRI"'"AL GIORNO"'
 540 IF ?#80=CH"R" GOTO 200
 550 END
 600 FOR I=1 TO S; PRINT$32; NEXT
 601 PRINT$255; FOR I=S+1 TO D; PRINT$9; NEXT; PRINT$255
 605 RETURN
 620 S=S-1; D=D-1; IF S=0 S=1; D=D+1
 625 GOTO 600
 640 S=S+1; D=D+1; IF D=30 D=29; S=S-1
 645 GOTO 600
 700 PRINT$#D
 703 @=1
 705 FOR J=9 TO 0 STEP -1; PRINT J
 707 I=1
 710 DO WAIT; I=I+1; UNTIL I=A;PRINT$8; NEXT
 715 PRINT$#D; H=H+1
 720 RETURN
 800 CLEAR0; MOVE0,0; DRAW10,20; DRAW24,30; DRAW32,48
 805 MOVE12,0; DRAW22,20; DRAW36,30; DRAW44,48
 810 MOVE27,30; PLOT5,26,29; MOVE28,30; PLOT5,27,29
 815 PRINT"         RALLY"''
 820 RETURN
1000 PRINT$#D
1010 FOR I=1 TO C; PRINT$9; NEXT; PRINT $255; RETURN
1400 DIM P(-1)
1500[:AA0 JSR #FFE3
1550 STA #80
1600 RTS
1650]
1700 RETURN
1800 END
```

Tastiera d'organo

Computer: Apple II	**Autore:** Rob Hausman
Modello: Europlus	**Versione italiana:** Pietro Canevarolo
Configurazione: 32 K	**Note:**

Mio figlio di un anno ha scoperto recentemente che premendo il tasto RESET del mio Apple II questo emette un sonoro bip.

Dopo dieci minuti di bip-bip, avevo deciso che era meglio se gli trovavo qualcosa di più serio e, soprattutto, variato con cui divertirsi. È nato così questo programma in linguaggio macchina per suonare sulla tastiera dell'Apple come fosse un organo.

Il programma è diviso in due parti: il vero e proprio programma che occupa le locazioni di memoria da $ 1000 a $ 1025 e una tabella di dati memorizzata da $ 1080 a $ 10FF.

I dati esadecimali possono essere inseriti in memoria sia con l'Integer Basic sia con il monitor dell'Applesoft.

Si ottiene ciò digitando CALL−151. Dovreste ottenere un * in risposta sullo schermo. Per esempio, i primi otto byte dei dati vengono memorizzati scrivendo semplicemente:

*1080: 1 1 1 1 1 1 1 1 –return–

Il numero che precede i due punti è l'indirizzo di partenza (esadecimale) e quelli che seguono sono i byte da memorizzare.

Lo stesso programma può essere scritto in memoria come un altro gruppo di dati, i codici-macchina e gli operandi esadecimali della ruotine.

Se avete l'Integer Basic, però, allora potete usate il mini-assembler in dotazione col linguaggio.

Per fare ciò, basta che scriviate:

*F666G –return–

Il punto esclamativo che compare sullo schermo è il "prompt" dell'assembler. Potete digitare ora la prima istruzione:

1000:LDA $ C030 –return–

Di nuovo, il numero che precede i due punti è l'indirizzo in cui va memorizzata l'istruzione.

Le istruzioni successive possono essere inserite semplicemente battendo uno spazio seguito dall'istruzione seguente. Mentre state scrivendo il programma premete il tasto RESET per ritornare al monitor e scrivete:

*1000L –return–

Otterrete così una lista del programma appena memorizzato.

A meno che non amiate riscrivere il programma ogni volta, salvatelo su nastro in questa maniera:

*1000.10FFW –return–

Potrete in seguito caricarlo dal registratore a cassette, scrivendo:

*1000.10FFR –return–

Per coloro che sono in possesso di un floppy-driver, la procedura è la seguente: dopo aver premuto RESET per ritornare al Basic, scrivete:

)BSAVE nnnn, A$ 1000, L$FF –return–

dove nnnn è il nome che volete utilizzare per questo programma.

Per caricare il programma dal disco, scrivete:

)BRUN nnnn –return–

con nnnn, ovviamente, uguale a quello utilizzato per il salvataggio. In ogni caso il programma si fa partire, col monitor, digitando:

*1000G –return–

Il programma, in effetti, è un unico ciclo. Ogni volta che viene eseguita la prima istruzione, viene inviato un impulso all'altoparlante dell'Apple selezionando l'indirizzo $C030 (hex). La seconda istruzione carica il byte alla locazione $C000 (hex) nel registro X.

Questa è una speciale locazione della memoria dell'Apple che contiene il codice dell'ultimo tasto che è stato premuto nei sette bit meno significativi e segnala, con l'ottavo bit, se il tasto è stato premuto dopo l'ultima volta che questa locazione di memoria è stata letta. Se il tasto è stato premuto questo bit è uguale a uno, altrimenti è zero. L'istruzione:

$1006 BPL $1019

esegue un test sul bit più significativo della locazione $C000, segnalando appunto se il tasto è stato premuto di recente oppure no.

Se la condizione appena esaminata risulta vera, il programma prosegue con l'istruzione di indirizzo $1008 che mette questo bit a zero.

L'istruzione seguente ($100B) seleziona un dato dalla tabella sommando a $1000 il contenuto del registro X. Questo si chiama indirizzamento indicizzato, dove X è l'indice che punta al byte cercato nella tabella. Il dato viene poi trasferito nella locazione $1030.

Le istruzioni di indirizzo $1011 fino a $1018 sono due cicli di ritardo concatenati. Consideriamo per primo il ciclo più interno:

$1013 DEY
$1014 BNE $1013

La prima istruzione toglie 1 al registro Y. Se il suo valore era 0, esso diventa uguale a 255. La seconda istruzione del ciclo fa ritornare il programma a $1013 fino a che il valore di Y è diverso da zero.

Il ciclo più interno consiste di tre istruzioni agli indirizzi $1011, $1016 e $1017. L'istruzione:

$1011 LDX $0F

carica il registro X con il valore 15 ($0F hex). Ciò significa che il registro deve essere decrementato 15 volte prima che il ciclo esterno sia terminato e il programma possa continuare alla istruzione in $1019.

Le restanti istruzioni creano il ritardo tra gli impulsi mandati all'altoparlante. Questo ritardo determina l'altezza della nota prodotta; più grande è il tempo di ritardo, più bassa è la nota. L'istruzione

$1019 LDX $1030

carica il registro X con il valore corrispondente alla nota suonata che è il numero di volte che deve essere eseguito il ciclo che segue nelle istruzioni da $101C a $1021. Il codice mnemonico NOP non fa compiere nessuna azione al computer e serve unicamente per far trascorrere un intervallo di tempo noto con precisione.

L'ultima istruzione rimanda il programma all'inizio per ricominciare la routine da capo.

Così come è l'organo usa solo i tasti della tastiera che sono indicati in figura. Se volete cambiare il numero di note possibili o i tasti usati, dovete semplicemente cambiare i valori della tabella dei dati. Per trovare il byte associato con un determinato tasto, sommate il codice ASCII di quel tasto a $1080 (hex.). Il risultato è l'indirizzo in cui dovrete memorizzare il valore associato alla nota corrispondente.

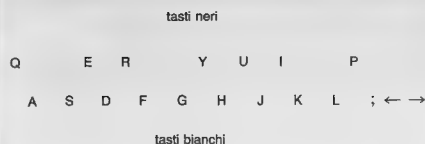



Fig. 1.

```
*1000.1029L

1000-    AD 30 C0     LDA   $C030
1003-    AE 00 C0     LDX   $C000
1006-    10 11        BPL   $1019
1008-    8D 10 C0     STA   $C010
100B-    BD 00 10     LDA   $1000,X
100E-    8D 30 10     STA   $1030
1011-    A2 0F        LDY   #$0F
1013-    88           DEY
1014-    D0 FD        BNE   $1013
1016-    CA           DEX
1017-    D0 FA        BNE   $1013
1019-    AE 30 10     LDX   $1030
101C-    CA           DEX
101D-    EA           NOP
101E-    EA           NOP
101F-    EA           NOP
1020-    EA           NOP
1021-    D0 F9        BNE   $101C
1023-    4C 00 10     JMP   $1000
1026-    4C 3A 94     JMP   $943A
*1030

1030- 76
*1080.10FF

1080- 01 01 01 01 01 01 01 01
1088- 39 01 01 01 01 01 01 01
1090- 01 01 01 01 01 36 01 01
1098- 01 01 01 01 01 01 01 01
10A0- 01 01 01 01 01 01 01 01
10A8- 01 01 01 01 01 01 01 01
10B0- 01 01 01 01 01 0  01 01
10B8- 01 01 01 41 01 01 01 01
10C0- 01 9F 01 01 85 80  6 6F
10C8- 63 52 58 4E 49 01 01 01
10D0- 45 A9 7D 96 01 5D 01 01
10D8- 01 69 01 01 01 01 01 01
10E0- 01 01 01 01 01 01 01 01
10E8- 01 01 01 01 01 01 01 01
10F0- 01 01 01 01 01 01 01 01
10F8- 01 01 01 01 01 01 01 01
```

Othello

Computer: Atari

Modello: 800

Configurazione: 16 K

Autore:

Versione italiana: Pietro Canevarolo

Note:

Ecco un adattamento di un popolare gioco di strategia chiamato Othello (o Reversi). In questa versione il giocatore si batte contro il computer e contro il tempo, scandito dal cronometro interno alla macchina. Rispetto alla solita maniera di introdurre i dati (ossia le coordinate della pedina da muovere), che di solito faceva uso della tastiera, in questa versione viene usato uno dei quattro joistick dell'Atari per introdurre le coordinate da giocare.

Anche la strategia, usata dal computer per valutare la sua posizione, è stata modificata, poiché i metodi tradizionali del Basic, richiedendo molti calcoli per prevedere le conseguenze di ogni mossa ed effettuare l'analisi di tutte le mosse possibili da una certa posizione, avrebbero richiesto troppo tempo rendendo il gioco troppo lento. Il programma valuta, invece, la sua posizione attuale sulla scacchiera in modo globale, anziché determinando il valore dei singoli pezzi. Il vantaggio è che il computer può scegliere la mossa da giocare in pochi secondi. D'altra parte, questa strategia non produce un gioco molto brillante.

Per renderlo più interessante è stato aggiunto un cronometro del tempo di gioco. In questo modo, lo scopo non è solo battere il computer, ma riuscirci nel minor tempo possibile. Si possono anche organizzare dei tornei tra amici per vedere chi è il vincitore più rapido. Ma non dovete sottovalutarlo! La strategia della macchina basta a battere severamente un giocatore non molto esperto. Perciò se non avete mai giocato prima ad Othello, dovete imparare a giocare e tentare poi di vincere la macchina. Anche dopo che la vostra esperienza sarà cresciuta, il computer vi batterà sonoramente se la vostra strategia si dimostra debole o se vi distraete nel corso della partita.

Per usare il programma, inserite un joistick nella presa numero uno. I pezzi del computer sono rossi, i vostri sono verdi. La scelta del giocatore che muove per primo viene effettuata tramite sorteggio dal computer. Due barre alla sinistra della scacchiera indicano il numero relativo dei pezzi dei due giocatori. Il numero esatto di pezzi di ciascuno e il tempo di esecuzione del giocatore umano sono mostrati nella finestra inferiore del video, riservata al testo.

Nel gioco originale, le pedine sono dei dischetti bicolori: una faccia bianca e una nera.

Il gioco consiste nel rovesciare le pedine dell'avversario circondandole con le proprie. Le pedine che vengono rovesciate sono quelle che si trovano comprese in linea retta (orizzontale, verticale o diagonale) tra la pedina che si sta giocando e le pedine dello stesso colore già sulla scacchiera.

Se al proprio turno non si possono rovesciare pedine avversarie, si perde la possibilità di giocare, e il gioco passa all'avversario. Vince chi, alla fine del gioco, ha il maggior numero di pedine del proprio colore.

Per giocare al vostro turno tirate la leva del joistick in una delle quattro direzioni per spostare il cursore nero sulla casella dove volete giocare la vostra mossa e, una volta posizionato esattamente il cursore, premete il pulsante rosso del joistick. Se tentate di effettuare una mossa non ammessa per tre volte di seguito, il computer crederà che non abbiate mosse legittime a vostra disposizione e si prenderà il turno di gioco. Il programma non riconosce la fine del gioco, perciò, alla fine, premete per tre volte di seguito il pulsante rosso, e il computer, rendendosi conto di non poter muovere, dichiarerà il vincitore. Buona fortuna e buon divertimento!

```
1 REM ********************************************
2 REM *                                          *
3 REM *              OTHELLO                     *
4 REM *           VERSIONE ATARI                 *
5 REM *                                          *
6 REM *          PERSONAL SOFTWARE               *
7 REM *                                          *
8 REM ********************************************
9 REM
10 GRAPHICS 5:SETCOLOR 4,10,6
12 SETCOLOR 0,4,12:SETCOLOR 1,14,12
14 POKE 752,1:SETCOLOR 2,9,2
20 DIM B(99),P(99),D(8),F(20)
22 M1=0:M2=0:M3=0
50 COLOR 3:FOR I=10 TU 66 STEP 7
51 PLOT I,0:DRAWTO I,39:NEXT I
52 FOR I=0 TO 35 STEP 5
54 PLOT 10,I:DRAWTO 66,I:NEXT I
60 FOR I=1 TO 8:READ X:D(I)=X:NEXT I
62 DATA -11,-10,-9,-1,1,9,10,11
64 FOR I=0 TO 99:B(I)=0:NEXT I
70 FOR I=0 TO 9:B(I)=9:B(I+90)=9:NEXT I
72 FOR I=1 TO 8:B(I*10)=9:B(I*10+9)=9:NEXT I
74 B(44)=1:B(45)=2:B(54)=2:B(55)=1
75 S=44:COLOR B(S):GOSUB 900:S=55:GOSUB 900
```

```
76 S=45:COLOR B(S):GOSUB 900:S=54:GOSUB 900
80 FOR I=1 TO 4:FOR J=1 TO 8:READ X:K=I*10+J
81 P(K)=X:P((4-I)*20+K+10)=X:NEXT J:NEXT I
91 DATA 9,2,8,6,6,8,2,9
92 DATA 2,1,3,4,4,3,1,2
93 DATA 8,3,7,5,5,7,3,8
94 DATA 6,4,5,0,0,5,4,6
289 POKE 18,M1:POKE 19,M2:POKE 20,M3
290 IF RND(0)<0.5 THEN 399
300 S=44:C=1:NM=0
302 POKE 18,M1:POKE 19,M2:POKE 20,M3
304 PRINT "Usa lo  stick  per muovere la pedina"
305 PRINT "nella casella che desideri ,"
306 PRINT "quindi premi il tasto di   FUOCO  "
310 X=(S-INT(S/10)*10)*7+4:Y=INT(S/10)*5-4
311 LOCATE X,Y,C
312 COLOR 3:PLOT X+1,Y+1:DRAWTO X+4,Y+1
313 PLOT X+1,Y+2:DRAWTO X+4,Y+2
320 IF STRIG(0)=0 THEN 350
322 T=STICK(0)
323 IF T=15 THEN SOUND 0,RND(0)*255,10,4:GOTO 312
324 COLOR C:PLOT X+1,Y+1:DRAWTO X+4,Y+1
325 PLOT X+1,Y+2:DRAWTO X+4,Y+2
330 IF T=7 THEN S=S+1:IF B(S)=9 THEN S=S-8
332 IF T=13 THEN S=S+10:IF S>90 THEN S=S-80
334 IF T=11 THEN S=S-1:IF B(S)=9 THEN S=S+8
336 IF T=14 THEN S=S-10:IF S<11 THEN S=S+80
345 GOTO 310
350 NM=NM+1:FOR I=1 TO 20:NEXT I
352 IF NM=3 THEN COLOR C:GOSUB 900:GOTO 399
355 IF B(S)<>0 THEN SM=0:COLOR C:GOSUB 900:GOTO 365
360 PM=2:PF=1:SM=0:GOSUB 800
365 IF SM<>0 THEN 370
367 PRINT :PRINT
368 PRINT "MOSSA NON CORRETTA":GOTO 3650
370 COLOR 3:GOSUB 900
375 N=1:F(1)=S:GOSUB 820
380 FOR K=1 TO N:B(F(K))=PM:NEXT K:GOSUB 850
382 FOR K=1 TO N:S=F(K):COLOR PM:GOSUB 900
383 FOR I=1 TO 30
385 SOUND 0,I,10,6:NEXT I:B(S)=PM:NEXT K
399 M1=PEEK(18):M2=PEEK(19):M3=PEEK(20):GOSUB 95
400 SQ=0:PF=2:PM=1
410 FOR S=11 TO 88
419 IF B(S)<>0 THEN 450
420 IF P(S)<P(SQ) THEN 450
421 SOUND 0,S*3,10,4
422 X=(S-INT(S/10)*10)*7+4
423 Y=INT(S/10)*5-4:LOCATE X,Y,C
424 COLOR 1:PLOT X+1,Y+1:DRAWTO X+4,Y+1
425 PLOT X+1,Y+2:DRAWTO X+4,Y+2
430 SM=0:GOSUB 800
440 IF SM=SQ AND RND(0)<0.5 THEN SQ=SM
442 IF SM>SQ THEN SQ=SM
445 COLOR C:PLOT X+1,Y+1:DRAWTO X+4,Y+1
446 PLOT X+1,Y+2:DRAWTO X+4,Y+2
450 NEXT S
451 IF SQ<>0 THEN 460
452 PRINT "NESSUNA MOSSA"
455 FOR I=1 TO 100:SOUND 0,150,10,4:NEXT I:GOTO 300
460 S=SQ:F(1)=SQ:N=1:GOSUB 820
470 FOR K=1 TO N:B(F(K))=PM:NEXT K
480 GOSUB 850
490 FOR K=1 TO N:S=F(K):COLOR PM:GOSUB 900
492 B(S)=PM:FOR I=1 TO 40:SOUND 0,I,10,8
493 NEXT I:NEXT K
495 GOTO 300
800 FOR J=1 TO 8:K=S+D(J):IF B(K)<>PF THEN 808
802 K=K+D(J):IF B(K)=PF THEN 802
804 IF B(K)<>PM THEN 808
806 SM=S:RETURN
808 NEXT J:RETURN
820 FOR J=1 TO 8:K=S+D(J):IF B(K)<>PF THEN 838
822 K=K+D(J):IF B(K)=PF THEN 822
824 IF B(K)<>PM THEN 838
825 K=K-D(J):X=(K-INT(K/10)*10)*7+4:Y=INT(K/10)*5-4
826 COLOR PM:PLOT X+1,Y+1:DRAWTO X+4,Y+1
827 PLOT X+1,Y+2:DRAWTO X+4,Y+2
828 IF K=S THEN 838
830 N=N+1:F(N)=K:GOTO 825
838 NEXT J:RETURN
850 CS=0:HS=0:COLOR 0
852 FOR I=1 TO 5:PLOT I,0:DRAWTO I,39:NEXT I
860 FOR S=11 TO 88
862 IF B(S)=1 THEN 870
864 IF B(S)=2 THEN 880
866 GOTO 890
870 CS=CS+1:C1=CS:IF C1>40 THEN C1=40
872 SOUND 0,CS*6,10,6:COLOR 1
873 PLOT 1,40-C1:PLOT 2,40-C1:GOTO 890
880 HS=HS+1:H1=HS:IF H1>40 THEN H1=40
882 SOUND 0,HS*6,10,6:COLOR 2
883 PLOT 4,40-H1:PLOT 5,40-H1
890 NEXT S:RETURN
900 X=(S-INT(S/10)*10)*7+4:Y=INT(S/10)*5-4
902 FOR I=Y TO Y+3:PLOT X,I
904 DRAWTO X+5,I:NEXT I:RETURN
950 PRINT " Computer   ";CS;" Giocatore  ";HS
952 S=INT((M3+M2*256+M1*65536)/60):M=INT(S/60)
953 S=S-INT(S/60)*60
954 PRINT :PRINT ,"Min ";M,"Sec ";S;"   ";
960 IF HS+CS<64 THEN ? :RETURN
965 FOR I=1 TO 999:NEXT I
970 PRINT :PRINT "FINE GIOCO"
971 PRINT :PRINT "PREMI FUOCO PER CONTINUARE";
990 SOUND 0,RND(0)*100,10,4
992 IF STRIG(0)=0 THEN RUN
994 GOTO 990
3650 SOUND 0,111,10,8
3652 FOR I=1 TO 700:NEXT I:GOTO 305
```

Se avete convertito uno dei nostri programmi per il vostro computer, spediteci il listato, una paginetta di spiegazioni e un supporto magnetico (disco o cassetta) con la registrazione del programma.

Vi restituiremo tre supporti magnetici e pubblicheremo la vostra versione.

Boing

Computer: Atari

Modello: 800

Configurazione: 16 K

Autore:

Versione italiana: Pietro Canevarolo

Note:

In questo gioco dovrete preoccuparvi della salute di un trampolinista un po' troppo esuberante! È il famoso rimbalzatore Bonzo.
Bonzo non rimbalza come le persone normali, perché le persone normali cercano di restare sul trampolino, mentre a Bonzo piace saltare in giro per tutta la palestra. Può rimbalzare sui muri senza nessun problema, ma sembra non rendersi conto che rimbalzare sul pavimento può essere pericoloso!
Il vostro impegno, dunque, è seguire Bonzo nelle sue giravolte con il trampolino, facendo in modo che non cada.

Sono previsti dieci livelli di difficoltà, da zero a nove, che dipendono da quanto esuberante si sente Bonzo in quel momento.

Ma, attenzione! Il livello nero è così facile, che potreste essere tentati di passare direttamente al livello nove. Non fatelo! Il livello nove è per persone che non hanno molto a cuore la salute del nostro pazzo Bonzo. Provate i livelli cominciando dal primo, cercando di controllare perfettamente il livello inferiore prima di passare al successivo.

Questa sarà tutta salute, per Bonzo!

```
1 REM ******************************************
2 REM *                                        *
3 REM *                 BOING                  *
4 REM *            VERSIONE ATARI              *
5 REM *                                        *
6 REM *           PERSONAL SOFTWARE            *
7 REM *                                        *
8 REM ******************************************
9 DIM B$(10):FOR I=1 TO 10:READ X:B$(I)=CHR$(X)
10 NEXT I:GRAPHICS 0:POKE 82,5:POSITION 9,1
15 PRINT "* * * B O I N G ! * * * ":PRINT
20 PRINT "Non far cadere BONZO rimbalzante!"
25 PRINT "A BONZO piace rimbalzare, ma ha"
30 PRINT "problemi a centrare il trampolino."
35 PRINT "Usa il joystick per spostarglielo."
45 PRINT "Se colpisce la parte sinistra del"
50 PRINT "trampolino rimbalza verso destra e"
55 PRINT "viceversa. Questo influisce sui"
70 PRINT "rimbalzi di BONZO sempre piu'"
75 PRINT "aumentando livello di difficolta'"
80 PRINT "e diventa sempre piu' difficile"
85 PRINT "mantenerlo in aria. Vediamo quante"
90 PRINT "volte riesci a far saltare BONZO."
95 PRINT :PRINT "        *** BUONA FORTUNA! ***   "
100 PRINT "       (BONZO ne ha bisogno!)"
105 PRINT
110 OPEN #1,4,0,"K"
120 PRINT "Livello di difficolta' (0-9)? ";
125 GET #1,Q:Q=Q-48
130 IF Q<0 OR Q>9 THEN 125
132 PRINT Q
140 H=(Q+2)/4:GRAPHICS 5:POKE 752,1
145 SETCOLOR 4,10,2:SETCOLOR 2,4,2
150 SETCOLOR 0,4,2:SETCOLOR 1,8,12
155 COLOR 1:FOR I=0 TO 5
157 PLOT I,39:DRAWTO I,0:NEXT I
160 DRAWTO 74,0:FOR I=74 TO 79
162 PLOT I,0:DRAWTO I,39:NEXT I
170 E=1.5:P=16:A=39:B=A:D=E:COUNT=1:V=14
172 C=INT(RND(0)*2+1)*H-3/2*H
175 POKE 656,0:POKE 657,P:PRINT B$;
180 FOR X=0 TO 16:FOR I=15 TO 32 STEP 17
182 POKE 656,0:POKE 657,X:PRINT CHR$(I);
185 FOR J=1 TO 10:NEXT J:NEXT I:NEXT X
190 POKE 657,16:PRINT CHR$(11);:FOR I=1 TO 50
192 NEXT I:POKE 657,16:PRINT " ";:FOR X=34 TO 39
200 COLOR 2:PLOT X,B
205 FOR I=1 TO 10:NEXT I:COLOR 0:PLOT X,B
210 FOR I=1 TO 50:NEXT I:NEXT X:COLOR 2:PLOT A,B
220 S=STICK(0)
225 IF S>8 AND S<12 AND P>2 THEN P=P-1
230 IF S>4 AND S<8 AND P<30 THEN P=P+1
235 SOUND 0,150+2*B,10,V:V=V-2:IF V<0 THEN V=0
240 POKE 656,0:POKE 657,P
242 IF B>38.5 THEN PRINT " BOING!";:GOTO 250
245 PRINT B$;
250 COLOR 0:PLOT INT(A),INT(B):A=A-C:B=B-D
255 D=D-0.4
260 IF A<6 THEN A=6:C=-C:GOSUB 400
265 IF A>73 THEN A=73:C=-C:GOSUB 400
270 IF B<1 THEN B=1:D=-ABS(D):E=E-RND(0)-1:GOSUB 400
275 IF B<38.5 THEN 320
280 B=39:D=E:V=14:LEFT=P+P
285 IF INT(A)>LEFT AND INT(A)<LEFT+14 THEN 310
290 PRINT
291 PRINT ,"SPLAT!",COUNT;"  Salt";
292 IF COUNT>1 THEN PRINT "i"
293 IF COUNT=1 THEN PRINT "o"
295 COLOR 2:PLOT INT(A),INT(B)
298 FOR V=14 TO 0 STEP -0.1:SOUND 0,253,12,V:NEXT V
300 PRINT "Il livello di difficolta' era ";Q
302 GOTO 120
310 IF B =38.5 THEN 320
312 COUNT=COUNT+1:E=E+RND(0)/2
314 C=(A-8-LEFT+RND(0))*H/4
320 COLOR 2:PLOT INT(A),INT(B):GOTO 220
999 DATA 32,13,13,13,13,13,13,32,32,42
```

Baseball

Computer: Atari

Modello: 800

Configurazione: 16 K

Autore:

Versione italiana: Pietro Canevarolo

Note:

Siamo in autunno. Siamo, cioè, nel bel mezzo dei vari campionati di calcio, pallacanestro, pallavolo, e altri sport. In America, siamo in pieno campionato di baseball, il loro sport nazionale che sta incontrando sempre maggior favore anche da noi.
Qui, a Personal Software, siamo in situazione di stallo nel nostro personale campionato di baseball tra colleghi di lavoro.
Ovviamente, vista la scarsità di spazi adatti al gioco vero e proprio e vista anche la nostra totale ignoranza del gioco, il nostro è un campionato computerizzato (occorreva dirlo?).
Il gioco originariamente è stato scritto per il TRS-80, e successivamente adattato all'Atari. Questa versione richiede l'uso dei joistick numero due e tre. Lo scopo è ovviamente segnare più punti possibili all'avversario. Il gioco si svolge tra due giocatori (che controllano le due squadre): i Rossi e i Blu. Per iniziare, dopo aver battuto il programma (e averne fatta una copia di emergenza), occorre connettere i due joistick e lanciare il programma.
Lo schermo vi informerà sulla squadra alla battuta.
La squadra che gioca in ricezione deve, naturalmente, ribattere la palla e correre per il campo per prendere la pallina prima che tocchi terra.
Per lanciare bisogna premere il pulsante di fuoco, oppure spostare la leva del joistick. A seconda della direzione scelta con il joistick, la palla volerà con diverse traiettorie e con vari effetti. Comunque la pratica è la miglior descrizione. Tirando la leva verso di sé si ottiene una palla diritta e veloce, mentre spingendola verso l'esterno la palla sarà più lenta e più ad effetto. Per colpire la palla si utilizza ancora la leva (dell'altro joistick, ovviamente). Anche qui, a seconda di come si muove la leva, si ottengono effetti vari: dal tiro diritto, fino al classico tiro "a campanile".

Una volta colpita la palla, viene il turno della corsa. Un giocatore della squadra del lanciatore deve partire di corsa per prendere la palla al volo, mentre il battitore corre intorno al campo per conquistare le varie basi. Quando la palla è stata raccolta, essa viene tirata verso i compagni di squadra che difendono le basi. Se il giocatore che difende una base riceve la palla prima che il battitore avversario sia arrivato a toccare la base, questo deve tornare indietro alla base precedente. Quando il giocatore che deve prendere la palla al volo è sullo schermo, dovete attendere che smetta di lampeggiare e premere il pulsante di fuoco. Il giocatore farà un incredibile salto e... tenterà di prendere la palla. Se la palla è troppo lunga, premete il pulsante finché il giocatore corre per farlo "lanciare" sulla palla. Sembra semplice, vero?

Il computer tiene conto del punteggio e mostra anche alcuni dati statistici (numero di battute, numero di errori, percentuale di errori) al termine di ogni *inning*.

Buon divertimento.

```
1 REM ***********************************************
2 REM *                                             *
3 REM *                 BASEBALL                    *
4 REM *              VERSIONE ATARI                 *
5 REM *                                             *
6 REM *            PERSONAL SOFTWARE                *
7 REM *                                             *
9 REM ***********************************************
80 DIM A$(7),S(18),N(47):IN=1:NT=1
80 DIM A$(7),S(18),N(47):IN=1:NT=1
82 FOR I=1 TO 18:S(I)=0:NEXT I
90 FOR I=1 TO 47:READ X:N(I)=X:NEXT I
92 DATA 67,67,34,41,46,54,46,46,61,61,61
93 DATA 67,67,34,41,46,54,46,46,46,46,46,46
94 DATA 41,41,41,54,51,46,41,41,51,61,61,61
95 DATA 41,41,41,41,43,41,36,36,46,61,61,61
100 IF B=2 THEN B=1:R=2:GOTO 120
115 B=2:R=1
120 P=0:OT=0:J=0:K=0:L=0
200 ST=0:BL=0:P1=P:E=0:H=0:GOSUB 800
210 IF P<=P1 THEN 230
220 FOR II=1 TO P-P1:FOR I=1 TO 100
222 SOUND 0,140-I,14,12:NEXT I:NEXT II
230 IF H=0 THEN OT=OT+1
240 IF B=2 THEN 250
242 IF H<>9 THEN BA=BA+1
244 IF H>0 AND H<5 THEN BH=BH+1
246 IF E=1 THEN RE=RE+1:BH=BH-1
248 GOTO 290
250 IF H<>9 THEN RA=RA+1
252 IF H>0 AND H<5 THEN RH=RH+1
254 IF E=1 THEN BE=BE+1:RH=RH-1
290 IF OT<3 THEN 200
300 S(IN)=P
400 GRAPHICS 0:SETCOLOR 2,1,4
402 SETCOLOR 4,8,10:POKE 752,1
405 PRINT :PRINT "           ALLSTAR BASEBALL "
407 PRINT :PRINT
```

```
410 PRINT "INNING":PRINT
412 PRINT :PRINT " ROSSI ":PRINT :PRINT " BLU  "
415 FOR I=1 TO 9:POSITION I*3+9,4:PRINT I;:NEXT I
420 BS=0:RS=0:FOR I=1 TO IN
422 IF I/2=INT(I/2) THEN 430
424 RS=RS+S(I):POSITION (I/2)*3+10,7
426 PRINT S(I);:GOTO 440
430 BS=BS+S(I):POSITION (I/2)*3+9,9
432 PRINT S(I);
440 NEXT I
450 POSITION 1,14
451 PRINT "Punti:     Punti Battute Errori Batt.%"
452 POSITION 2,16
454 PRINT " Rossi ":POSITION 2,18:PRINT " Blu "
460 POSITION 13,16:PRINT RS:POSITION 19,16
461 PRINT RH:POSITION 26,16:PRINT RE
462 PE=INT(RH/RA*1000):POSITION 33,16:PRINT PE
464 POSITION 13,18:PRINT BS:POSITION 19,18
465 PRINT BH:POSITION 26,18:PRINT BE
466 IF BA=0 THEN 479
468 PE=INT(BH/BA*1000):POSITION 33,18:PRINT PE
470 POSITION 0,23
479 PRINT "Premi il pulsante per continuare...";
480 IF STRIG(1)=0 OR STRIG(2)=0 THEN GOTO 485
482 GOSUB 950:GOTO 480
485 SOUND 1,0,0,0:IN=IN+1
486 IF IN=19 THEN 491
487 IF IN=18 AND BS>RS THEN 491
490 GOTO 100
491 PRINT "Premi il pulsante per giocare ancora";
492 IF STRIG(1)=0 OR STRIG(2)=0 THEN RUN
493 GOSUB 950:GOTO 492
600 IF H<1 OR H>4 THEN RETURN
602 FOR I=1 TO H*2:SETCOLOR 4,RND(0)*16,12
603 SOUND 0,100,8,10:FOR II=1 TO 100:NEXT II
604 NEXT I
606 GOTO H*10+600
610 IF L=1 THEN L=0:P=P+1
612 IF K=1 THEN K=0:L=1
614 IF J=1 THEN K=1
616 J=1:RETURN
620 IF L=1 THEN L=0:P=P+1
622 IF K=1 THEN P=P+1
624 IF J=1 THEN J=0:L=1
626 K=1:RETURN
630 IF L=1 THEN P=P+1
632 IF K=1 THEN P=P+1:K=0
634 IF J=1 THEN P=P+1:J=0
636 L=1:RETURN
640 IF J=1 THEN J=0:P=P+1
642 IF K=1 THEN K=0:P=P+1
644 IF L=1 THEN L=0:P=P+1
646 P=P+1:RETURN
650 FOR I=1 TO 200:NEXT I
651 IF J=0 THEN J=1:RETURN
652 IF K=0 THEN K=1:RETURN
654 IF L=0 THEN L=1:RETURN
656 P=P+1:RETURN
660 PRINT " OUT! ":SETCOLOR 4,6,12:SOUND 0,100,6,8
661 FOR I=1 TO 200:NEXT I
662 IF J=0 OR OT>1 OR RND(0)<0.5 THEN RETURN
663 PRINT " BATTUTA DOPPIA! ":SOUND 0,200,10,8
664 FOR I=1 TO 250:NEXT I
665 IF OT=0 THEN 670
666 OT=OT+1:RETURN
670 OT=OT+1:IF K=1 AND L=0 THEN K=0:J=1:RETURN
672 IF L=0 AND K=0 THEN J=0:RETURN
674 IF L=1 AND K=0 THEN P=P+1:J=0:L=0:RETURN
676 J=0:RETURN
680 SETCOLOR 4,4,12:SOUND 0,100,6,8
681 FOR I=1 TO 100:NEXT I
682 IF OT>1 OR L=0 OR RND(0)<0.4 THEN RETURN
684 PRINT " RINUNCIA alla corsa! "
686 SETCOLOR 4,11,12:SOUND 0,200,6,8
687 FOR I=1 TO 200:NEXT I
688 P=P+1:L=0:OT=OT+1:H=9:RETURN
700 GRAPHICS 5:SETCOLOR 4,13,6:SETCOLOR 0,0,12
701 SETCOLOR B,9,8:SETCOLOR R,14,4:POKE 752,1:DV=0
702 IF RND(0)<0.5 THEN 750
703 PRINT " BATTUTA INTERNA!  ":PRINT
704 SETCOLOR 4,1,10
705 X=INT(RND(0)*25)*2+10:Y=3:COLOR 3:GOSUB 980
708 A=RND(0)*40+20:D=39:C=(RND(0)-0.5)
709 E=-(RND(0)+2)
720 COLOR 0:GOSUB 980
722 IF STICK(B)>4 AND STICK(B)<8 THEN X=X+2:GOTO 2000
723 IF STICK(B)>8 AND STICK(B)<12 THEN X=X-2:GOTO 2010
724 IF X<2 THEN X=2
725 IF X>74 THEN X=74
726 COLOR 3:GOSUB 980
728 IF A<3 OR A>76 THEN 745
730 LOCATE A,D-1,Z:IF Z=3 THEN GOTO 660
734 COLOR 0:PLOT A,D
735 A=A+C:D=D+E:COLOR 1:PLOT A,D
736 IF D<3 THEN GOTO 740
737 SOUND 0,D*5,10,5:IF DV>0 THEN 728
739 GOTO 720
740 IF ABS(X-A)>9 THEN GOTO 747
742 IF RND(0)<0.5 AND ABS(X+1-A)<5 THEN 3000
745 H=1:PRINT " SINGOLA ":GOTO 600
747 H=2:PRINT " DOPPIA ":GOTO 600
750 X=INT(RND(0)*25)*2+10:Y=34:COLOR 3:GOSUB 980
751 PRINT " BATTUTA ESTERNA! ":PRINT
752 A=INT(RND(0)*20):D=INT(RND(0)*5)
753 C=RND(0)+1:E=RND(0)+0.6
755 COLOR 1:PLOT 77,30:DRAWTO 77,39:PLOT A,D
760 COLOR 0:GOSUB 980
761 IF STRIG(B)=0 AND STICK(B)=15 THEN 4000
762 IF STICK(B)>4 AND STICK(B)<8 THEN X=X+1:GOTO 2020
763 IF STICK(B)>8 AND STICK(B)<12 THEN X=X-1:GOTO 2030
764 IF X<2 THEN X=2
765 IF X>74 THEN X=74
766 COLOR 3:GOSUB 980
767 IF A+C>76 AND D<29 THEN GOTO 785
769 IF A+C>76 THEN GOTO 786
770 LOCATE A,D+1,Z:IF Z=3 THEN 774
772 PRINT "OUT!!":SOUND 0,40,4,10:GOTO 680
774 COLOR 0:PLOT A,D
775 A=A+C:D=D+E:COLOR 1:PLOT A,D
776 IF D>37 THEN GOTO 790
780 SOUND 0,A+C*X+1,10,5:IF DV>0 THEN GOTO 767
783 GOTO 760
785 H=4:PRINT " FUORI CAMPO! ":GOTO 600
786 H=3:PRINT " TRIPLA ":GOTO 600
790 IF A>55 THEN GOTO 747
794 GOTO 742
800 GRAPHICS 5:SETCOLOR 4,13,6:SETCOLOR 0,1,10
801 SETCOLOR B,?,0:SETCOLOR R,14,4:POKE 752,1
805 PRINT " Punti  ";P;"        Outs      ";OT
806 PRINT " BALLS  ";BL;"        STRIKES  ";ST:PRINT
810 COLOR 1:PLOT 35,31:DRAWTO 4,0
812 PLOT 44,31:DRAWTO 75,0
814 Z=29:FOR I=55 TO 60:PLOT I,Z
815 DRAWTO 40,60-I:DRAWTO I,Z-1:Z=Z-1:NEXT I
817 Z=14:FOR I=17 TO 24:PLOT I,Z
818 DRAWTO 39,I-19:DRAWTO I,Z-1:Z=Z+1:NEXT I
820 COLOR 3:B1=[illegible]:B2=5:GOSUB 990
822 B1=22:B2=1 :GOSUB 990
824 B1=56:B2=17:GOSUB 990
825 PLOT 39,33:DRAWTO 41,33
826 PLOT 39,34:PLOT 41,34:PLOT 40,35
830 PLOT 38,10:DRAWTO 41,10
834 COLOR 2:PLOT 37,33:DRAWTO 37,37
840 IF J=0 THEN X=54:Y=11:GOSUB 980
842 IF K=0 THEN X=33:Y=2:GOSUB 980
844 IF L=0 THEN X=19:Y=19:GOSUB 980
850 Y=11:X=38+INT(RND(0)*4):SW=0
852 A$="ROSSI":IF B=2 THEN A$="BLU"
854 PRINT " LANCIANO I ";A$;"!  ";
855 S=STICK(B)
856 IF S=15 THEN SOUND 0,RND(0)*200,10,2:GOTO 855
858 S=S/6
860 COLOR 4:PLOT X,Y:SOUND 0,Y,10,8
862 IF STICK(B)=7 AND Y<33 THEN X=X+0.3:GOTO 5000
863 IF STICK(B)=11 AND Y<33 THEN X=X-0.3:GOTO 5010
865 Y=Y+S:LOCATE X,Y,Z:IF Z=2 THEN 890
866 LOCATE X,Y+1,Z:IF Z=2 THEN 890
867 LOCATE X,Y+2,Z:IF Z=2 THEN 890
870 COLOR 1:PLOT X,Y
871 IF SW>0 THEN GOSUB 900+SW:SW=SW+1:GOTO 873
872 IF STICK(R)<>15 THEN SW=1
873 IF Y<38 THEN 860
874 IF SW=1 AND (X<39 OR X>41) THEN 6000
875 ST=ST+1:SETCOLOR 4,8,10
876 SOUND 0,100,10,8:FOR I=1 TO 30:NEXT I
877 IF BL=4 THEN 884
```

```
878 IF ST=3 THEN 885
879 GOTO 800
884 PRINT "CAMMINATA":H=9:GOTO 7000
885 PRINT "STRIKE OUT":SETCOLOR 4,8,10
886 FOR I=1 TO 100:SOUND 0,200,10,8:NEXT I:RETURN
890 D=RND(0):IF RND(0)<0.5 THEN D=-D
891 SOUND 0,S,4,14
892 COLOR 0:PLOT X,Y:X=X+D:Y=Y-1
894 IF Y<1 THEN 700
896 COLOR 1:PLOT X,Y:SOUND 0,X+2*Y,4,14:GOTO 892
901 COLOR 4:PLOT 37,33:DRAWTO 37,37:GOTO 9010
902 COLOR 4:PLOT 37,33:DRAWTO 41,37:GOTO 9020
903 COLOR 4:PLOT 37,33:DRAWTO 42,33:GOTO 9030
904 COLOR 4:PLOT 37,33:DRAWTO 41,29:RETURN
905 SW=4:COLOR 2:PLOT 37,33:DRAWTO 37,29:RETURN
950 SOUND 0,N(NT),10,8
952 FOR I=1 TO 10:NEXT I
954 NT=NT+1:IF NT>47 THEN NT=1
956 RETURN
980 PLOT X+1,Y:PLOT X,Y+1
981 DRAWTO X+2,Y+1:PLOT X+1,Y+2
982 PLOT X,Y+3:PLOT X,Y+4
983 PLOT X+2,Y+3:PLOT X+2,Y+4:RETURN
990 PLOT B1,B2:PLOT B1+1,B2
991 PLOT B1,B2+1:PLOT B1+1,B2+1:RETURN
992 COLOR 3:X=X-RND(0)*9:IF X<1 THEN X=1
993 PLOT X,Y+1:PLOT X+1,Y+2:DRAWTO X+1,Y+4:GOTO 9930
994 PLOT X+3,Y+4:PLOT X+4,Y+4:RETURN
995 COLOR 3:X=X+RND(0)*9:IF X>70 THEN X=70
996 PLOT X,Y+2:PLOT X+1,Y+2:GOTO 9960
997 PLOT X,Y+4:PLOT X+1,Y+4:RETURN
2000 IF STRIG(B)=0 THEN DV=1:GOSUB 995:GOTO 728
2010 IF STRIG(B)=0 THEN DV=1:GOSUB 992:GOTO 728
2020 IF STRIG(B)=0 THEN DV=1:GOSUB 995:GOTO 767
2030 IF STRIG(B)=0 THEN DV=1:GOSUB 992:GOTO 767
3000 E=1:H=1:PRINT " ERRORE #%@&''''"
3010 SOUND 0,150,10,6:GOTO 600
4000 Y=Y-2-RND(0)*6:DV=1:GOTO 766
5000 IF X>43 THEN X=43:GOTO 863
5010 IF X<38 THEN X=38:GOTO 865
6000 BL=BL+1:SETCOLOR 4,12,10:SOUND 0,200,6,4
6010 FOR I=1 TO 30:NEXT I:GOTO 877
7000 SETCOLOR 4,12,10:SOUND 0,100,6,8:GOTO 650
9010 COLOR 2:PLOT 37,33:DRAWTO 41,37:RETURN
9020 COLOR 2:PLOT 37,33:DRAWTO 42,33:RETURN
9030 COLOR 2:PLOT 37,33:DRAWTO 41,29:RETURN
9930 PLOT X,Y+3:PLOT X+2,Y+3:PLOT X+3,Y+2
9932 PLOT X+4,Y+2:GOTO 994
9960 PLOT X+2,Y+3:DRAWTO X+4,Y+3:PLOT X+3,Y+2
9962 PLOT X+3,Y+4:PLOT X+4,Y+1:GOTO 997
```

Supercaccia

Computer: Commodore

Modello: VIC 20

Configurazione: 3 K

Autore: Pietro Canevarolo

Note:

Supercaccia è un rifacimento per il VIC-20 Commodore di un gioco molto diffuso nei bar e nelle sale giochi. Il vostro compito è mangiare i tesori che appaiono in un labirinto prima che il mostro mangi voi. Sembra semplice, vero? Be', non è proprio così facile come sembra. Più veloci siete, maggiore sarà anche la velocità con cui si muove il mostro. A che scopo, allora, correre velocemente? Il fatto è che più velocemente vi spostate, maggiore è il punteggio per ogni tesoro catturato.

Ecco come funziona il gioco. Per prima cosa dovete scegliere il livello di difficoltà. I livelli disponibili sono nove, da 1 a 9. Dopo che avrete effettuato la vostra scelta, il programma disegna il labirinto. Finita anche questa operazione, vengono sistemati i tesori che dovrete mangiare e il simbolo che vi rappresenta appare nell'angolo a sinistra in alto dello schermo. A questo punto potete partire.

Se riuscite a ripulire tutto il labirinto dai tesori, vi viene assegnato un premio, che dipende dal livello di difficoltà che avete scelto in precedenza, poi viene disegnato un altro labirinto e il gioco ricomincia con un livello di difficoltà superiore a quello precedente. Non preoccupatevi se non riuscite a ottenere un punteggio molto alto alle prime partite. Di solito si impara abbastanza in fretta.

Il mostro seguirà il vostro percorso passo per passo, così potete nascondervi in un corridoio laterale del labirinto e lasciare che vada avanti se vi ricordate la strada che avete percorso. Se vi trovate in trappola, cercate di farlo accelerare. Mentre il mostro sta accelerando, voi potete correre e sorpassarlo. Per ottenere questo effetto, dovete muovervi avanti e indietro più in fretta che potete.

Alla fine, se il mostro vi ha catturato, appare il punteggio e il tempo che siete riusciti a sopravvivere.

```
1 REM ********************************************
2 REM *                                          *
3 REM *              SUPERCACCIA                 *
4 REM *            VERSIONE VIC-20               *
5 REM *                                          *
6 REM *          PERSONAL SOFTWARE               *
7 REM *                                          *
8 REM ********************************************
40 GOSUB14000
42 POKE1,0 POKE2,0
45 GOSUB12000 CLR SK=PEEK(0) P=PEEK(1)*256+PEEK(2)
100 GOTO10000
1000 M$=""
1110 POKEDD,127 P1=PEEK(D1)ANDAD P2=PEEK(D2)
1120 IFP1<>58THEN1130
1122 M$="+▮◘" PRINTM$, Y=Y-1
1124 C$="◙" CX=0 CY=1 GOTO1160
1130 IFP2<>119THEN1140
1132 M$="+▮▮" PRINTM$  X=X+1
1134 C$="▮" CX=-1 CY=0 GOTO1160
1140 IFP1<>46THEN1150
1142 M$="+▮▮" PRINTM$, X=X-1
1144 C$="▮" CX=1 CY=0 GOTO1160
1150 IFP1<>54THEN1155
1152 M$="+▮◙" PRINTM$, Y=Y+1
1154 C$="◘" CY=-1 CX=0 GOTO1160
1155 GOTO1300
1160 IFFNCH(S)<>WLTHEN1170
1162 PRINTC$, X=X+CX Y=Y+CY GOTO1300
1170 F$=F$+RIGHT$(M$,1)
1180 IFFNCH(S)=DITHENP=P+100*(EL-S):PC=PC+1
1190 IFFNCH(S)=SPTHENP=P+50*(EL-S) PC=PC+1
1200 IFFNCH(S)=CLTHENP=P+30*(EL-S) PC=PC+1
1210 IFFNCH(S)=HETHENP=P+20*(EL-S):PC=PC+1
1220 IFFNCH(S)=CITHENP=P+10*(EL-S) PC=PC+1
1250 J$=STR$(P*SK) FORJ=1TOLEN(J$)
1270 POKESC+J+489,ASC(MID$(J$,J,1)):NEXT
1300 PRINT"●▮",
1310 IFPC<61THEN1900
1312 PRINTDN$"IL TESORO NON C'E", GOTO7000
1900 RETURN
2000 IFLEN(F$)>=30THENGOSUB3000
2005 FM=FM+1 IFFM/S<>INT(FM/S)THENRETURN
2006 FORH=1TOSKL
2007 POKEFNPLOT(0),32
2010 J$=LEFT$(F$,1) F$=MID$(F$,2)
2030 POKEV,15 POKES1,254-LEN(F$)
2050 FORM=1TO10 NEXT POKE36875,0
2100 IFJ$="◘"THENYF=YF-1 GOTO2200
2110 IFJ$="▮"THENXF=XF+1 GOTO2200
2120 IFJ$="◙"THENYF=YF+1 GOTO2200
2130 IFJ$="▮"THENXF=XF-1 GOTO2200
2150 GOTO2200
2200 POKEFNPL(0),42
2205 NEXT
2210 RETURN
```

```
3000 POKEFNPL(0),32
3007 S=S-1:IFS<1THENS=1
3008 J$=STR$(EL-S):FORJ=1TOLEN(J$)
3009 POKESC+J+502,ASC(MID$(J$,J,1)):NEXT
3010 FORC=1TO10:J$=MID$(F$,C,1)
3015 IFJ$="]"THENYF=YF-1:GOTO3100
3020 IFJ$="@"THENYF=YF+1 GOTO3100
3030 IFJ$="█"THENXF=XF+1 GOTO3100
3040 IFJ$="█"THENXF=XF-1:GOTO3100
3100 POKEFNPL(0),42
3150 IFC/SK=INT(C/SK)THENGOSUB1000
3310 FORM=CTOC+2:POKES2,M*3+130
3320 FORN=1TO10 NEXT NEXT POKES2,0
3350 POKEFNPLOT(0),32
3400 NEXT
3500 F$=MID$(F$ EL) RETURN
4000 IFX<>XFORY<>YFTHEN4500
4100 PRINTDN$"OTTIMO BOCCONE!"GOSUB6000 GOTO11000
4500 RETURN
6000 POKE36877,220:FORL=15TO0STEP-1 POKE36878 L
6005 FORM=1TO300:NEXT NEXT POKE36877,0 POKE36878,15
6010 RETURN
7000 FORK=1TO30
7010 POKE36876,0:FORL=1TO5 NEXT NEXT
7100 J=INT(P/256) POKE1,J POKE2,P-J*256
7200 SK=PEEK(0)+1 POKE0,SK GOTO45
7999 GOTO7000
8000 FORM=1TO500:GOSUB1000 IFLEN(F$)<20THENNEXT
8010 FORJ=8142TO8142+20 POKEJ,32 NEXT
8100 GOSUB1000 GOSUB2000 GOSUB4000 GOTO8100
10000 DN$="█████████████████████"
10030 S=10 PC=0 SC=7680 RO=22
10050 DEFFNPL(XX)=(YF*RO+XF)+SC
10060 DEFFNCH(XX)=PEEK((Y*RO+X)+SC)
10077 SO=10:POKE36878,15
10100 DD=37154 D1=37151 D2=37152 AD=63
10110 WL=102 DI=90 SP=65 CL=88:HE=83 CI=87 EL=11
10120 V=36878 S1=36875 S2=36876
10500 TI$="000000"
10600 PRINTDN$"        PARTI!     "
10700 PRINTDN$"██PUNTI █     █VELOC:█ 1█"
10800 PRINT"███"::X=1:Y=1:XF=1:YF=1
10900 GOTO8000
11000 REM FINE GIOCO
11010 POKE37154,255
11105 PRINT:PRINT"█TEMPO ";MID$(TI$,3,2),
11107 PRINT" MI E ";RIGHT$(TI$,2);" SEC"
11110 PRINT"ANCORA? $█";
11120 GETJ$:IFJ$=""THEN11120
11130 PRINTJ$:IFJ$="N"THENEND
11140 IFJ$="S"THENRUN
11150 PRINT:PRINT"]"; GOTO11110
12000 DIMA(3):A(0)=2 A(1)=-44 A(2)=-2:A(3)=44
12005 WL=102:HL=32:SC=7680 A=SC+23 J=RND(-TI)
12010 POKE36879,111
12100 PRINT"██",CHR$(142);:FORI=1TO21
12110 PRINT"██████████████████████" NEXT POKEA,4
12200 J=INT(RND(1)*4) X=J
12205 B=A+A(J)
12210 IFPEEK(B)<>WLTHEN12240
12220 POKEB,J:POKEA+A(J)/2,HL:A=B GOTO12200
12240 J=(J+1)*-(J<3):IFJ<>XTHEN12205
12250 J=PEEK(A):POKEA,HL IFJ<4THENA=A-A(J):GOTO12200
12300 PRINT"███X"
12305 READJ,K,C:IFJ<0THEN12500
12310 FORA=1TOJ
12320 B=INT(RND(1)*410)
12325 IF(B-21)/22=INT((B-21)/22)THEN12320
12330 IFPEEK(B+7702)<>32THEN12320
12340 POKEB+7702,K:POKEB+38422,C
12350 NEXT:GOTO12305
12400 DATA2,90,1,4,65,0,7,88,5,9,83,2,39,87,3,-1,0,0
12450 PRINT"12450 P="P
12500 RETURN
14000 POKE36879,46
14010 PRINTCHR$(14)"]██████     █UPER-ACCIA█"
14020 PRINT"████ TI TONY TODSHALL"
14050 PRINT"███ ABILITA'  (1-9)   █ ███",
14060 GETJ$ IFJ$=""THEN14060
14070 SK=VAL(J$):IFSK<1ORSK>9THEN14060
14075 POKE0,SK
14080 PRINTSK POKE0 SK FORN=1TO500 NEXT RETURN
```

Tira e molla

Computer: Sinclair

Modello: ZX 81

Configurazione: 16 K

Autore: Max Huber

Versione italiana: Pietro Canevarolo

Note:

Lo scopo del gioco è di far atterrare una nave spaziale sopra una piattaforma, usando i tasti 5 e 8 per spostarsi lateralmente e il tasto 7 per mantenersi in aria.

La piattaforma è disegnata utilizzando il tasto SHIFT-H dopo aver premuto GRAPHICS, ossia SHIFT-9. Il punteggio viene visualizzato sotto alla piattaforma. Ovviamente se state sospesi in aria consumate carburante e, all'esaurimento di questo, perdete il controllo dell'astronave. Ciò avviene anche se mancate il bersaglio. Il gioco ricomincia premendo lo zero dopo che sarà apparso il punteggio.

L'animazione grafica viene effettuata sovrapponendo rapidamente la stampa dell'astronave nella posizione attuale alla vecchia posizione. Il messaggio di PERICOLO lampeggiante e della quantità di carburante utilizza la stessa tecnica.

Per aumentare la difficoltà del gioco, provate la seguente modifica:

520 LET X=INT (RND*5)+6

Provate anche altri valori per le due costanti e fateci sapere come funziona il gioco.

Il programma occupa circa 4K di memoria e, quindi, non gira su macchine non espanse.

Il carattere grafico nella riga 300 è una S inversa mentre alla riga 530 viene utilizzato il carattere SHIFT-H.

```
   1 REM TIRA E MOLLA
   3 LET SC=0
   4 LET S=20
  10 GOSUB 500
  15 PRINT AT 0,0;"CARBURANTE=";
S;"  "
  17 PRINT AT 1,0;"PUNTI=";SC
  20 PRINT AT Y,X;"     "
  30 PRINT AT Y+1,X;" <*> "
  40 IF INKEY$="5" THEN LET X=X-
1
  50 IF INKEY$="8" THEN LET X=X+
1
  60 IF INKEY$="7" THEN GOTO 300
  70 IF X<=-1 THEN LET X=0
  80 IF X>=28 THEN LET X=28
  90 LET Y=Y+1
 100 IF Y=17 THEN GOTO 400
 110 GOTO 20
 300 PRINT AT Y+2,X;"  █  "
 310 LET S=S-2
 320 IF S=-2 THEN GOTO 700
 330 GOTO 15
 400 IF NOT (X=1 OR X=13 OR X=27
) THEN GOTO 700
 410 IF X=1 THEN LET SC=SC+200
 420 IF X=13 THEN LET SC=SC+100
 430 IF X=27 THEN LET SC=SC+150
 440 PRINT AT 10,10;"BEN FATTO"
 450 PAUSE 120
 460 CLS
 470 GOTO 10
 500 PRINT AT 21,0;"[illegible]"
 510 LET X=INT (RND*27)+1
 520 LET Y=INT (RND*5)+3
 530 PRINT AT 18,0;"[illegible]"
 540 PRINT AT 19,0;"[illegible]"
 550 PRINT AT 20,0;"[illegible]"
 560 RETURN
 700 PRINT AT Y,X+1;"   "
 710 PRINT AT Y+1,X+1;"<*>"
 720 LET Y=Y+1
 730 IF Y=21 THEN GOTO 800
 740 PRINT AT 10,10;"[illegible]"
 750 FOR N=1 TO 10
 760 NEXT N
 770 PRINT AT 10,10;"[illegible]"
 780 GOTO 700
 800 CLS
 810 PRINT AT 10,8;"MISSIONE FAL
LITA"
 815 PRINT AT 12,10;"PUNTI=";SC
 820 FOR N=1 TO 10
 830 NEXT N
 840 IF INKEY$="0" THEN GOTO 900
 850 GOTO 800
 900 CLS
 910 RUN
```

Trappola

Computer: Sinclair ZX 81

Configurazione: 1 K

Autore: Max Huber

Versione italiana: Pietro Canevarolo

Un ragazzo di 14 anni ha inventato questo gioco per ZX81 (o ZX80 con ROM nuova da 8K) e 1K di RAM.

Il computer visualizza sullo schermo un asterisco (*), che potete spostare con i tasti a freccia, ossia 5, 6, 7 e 8, attraverso tutto lo schermo. Mentre voi vi muovete il computer cerca di intrappolarvi costruendo dei muri che sbarrano il vostro percorso utilizzando lo spazio inverso. (Per chi non lo sapesse ricordiamo che lo spazio inverso si ottiene premendo GRAPHICS, ossia SHIFT-9, e poi lo SPACE.) Voi dovete cercare di non farvi imprigionare dal computer e di non urtare i muri. I muri divengono un labirinto nel corso del gioco e il punteggio vi viene mostrato al momento dell'urto con un muro.

Qualche volta, se il programma viene fatto girare su macchine con solo 1K di memoria, il gioco si ferma e appare il messaggio 5/XXX, XXX è il numero di una riga, che significa che non c'è più posto disponibile sullo schermo. Il gioco riprende con il punteggio inalterato premendo il tasto CONT.

```
  5 REM *** TRAPPOLA ***
  7 CLS
 10 LET S=0
 20 LET A=7
 30 LET B=16
 40 PRINT AT A,B;"*"
 50 LET C=A
 60 LET D=B
 70 LET C=C+INT (RND*3-1)
 80 LET D=D+INT (RND*3-1)
 90 IF C=A AND D=B THEN GOTO 70
100 PRINT AT C,D;"█"
120 PRINT AT A,B;"*"
130 IF INKEY$="5" THEN LET B=B-
1
140 IF INKEY$="8" THEN LET B=B+
1
150 IF INKEY$="6" THEN LET A=A+
1
160 IF INKEY$="7" THEN LET A=A-
1
170 PRINT AT A,B;
180 IF PEEK (PEEK 16398+PEEK 16
399*256)=128 THEN GOTO 210
190 IF INKEY$<>"" THEN LET S=S+
1
200 GOTO 40
210 PRINT AT 18,1;"(";S;")"
220 FOR I=0 TO 300
230 NEXT I
240 RUN
```

Stemma

Computer: Sinclair ZX 81

Configurazione: 1 K

Autore: Max Huber

Versione italiana: Pietro Canevarolo

Il programma è stato concepito per girare sullo ZX81 con 1K di memoria e stampante ZX, oppure sullo ZX80 con ROM nuova da 8K e stampante.

Dopo che avrete dato il RUN, il computer vi chiederà di indicargli la larghezza che desiderate abbia lo stemma. Il numero che dovete battere deva essere scelto tra 1, 2, 3 e 4. Più piccolo è il numero, più piccolo risulterà lo stemma disegnato sulla stampante. In seguito il computer chiederà il messaggio che desiderate appaia sullo stemma e, infine, lo stamperà.

Una versione migliorata del programma si ottiene con le seguenti modifiche:

5 INPUT W
170 FOR X=1 TO W

In questo caso il computer vi chiede per prima la larghezza, un numero da 1 a 9, poi l'altezza, da 1 a 4, e, infine, il messaggio dello stemma.

```
 10 INPUT S
 20 INPUT A$
 30 DIM A(8)
 40 LET C=CODE A$
 50 FOR X=1 TO 8
 60 LET A(X)=PEEK (7679+C*8+X)
 70 NEXT X
 80 LET D=128
 90 FOR A=1 TO 8
100 DIM B$(8)
110 FOR B=1 TO 8
120 IF A(B)<D THEN GOTO 150
130 LET B$(9-B)="█"
140 LET A(B)=A(B)-D
150 NEXT B
160 LET D=D/2
170 FOR X=1 TO S
180 FOR Y=1 TO 8
190 FOR Z=1 TO S
200 LPRINT B$(Y);
210 NEXT Z
220 NEXT Y
230 LPRINT
240 NEXT X
250 NEXT A
260 LET A$=A$(2 TO )
270 IF A$="" THEN RUN
280 GOTO 40
```

Pianeta X

Computer: Sinclair ZX 81

Configurazione: 1 K

Autore : Max Huber

Versione italiana: Pietro Canevarolo

Siete seduti al posto di guida di un'astronave (£) che entra ed esce da vallate e crateri, sorvola montagne e pianure, alla ricerca di minerali preziosi. Il computer di bordo effettua la ricerca e l'analisi automatica dei campioni di minerale sulla superficie del pianeta e il vostro compito si limita a mantenere l'astronave in volo più vicina possibile al suolo senza schiantarsi addosso a qualche rilievo del terreno.

Premendo il tasto 5 l'astronave si sposta verso sinistra, mentre con l'8 vi dirigete a destra. Attenzione, però! Più alto è il vostro punteggio, più lenta è l'azione dei razzi di direzione dell'astronave. Otterrete il punteggio più alto volando più rasente possibile al suolo. Questo vi verrà mostrato solo al momento dell'impatto col pianeta. Prima di iniziare a volare, dovete descrivere al computer la natura della superficie del pianeta, ossia dovete rispondere alla domanda del computer con un carattere che verrà utilizzato dal programma per disegnare il profilo orografico del pianeta.

```
   5 INPUT A$
   6 LET U=15
   7 LET P=15
   8 LET S=0
   9 LET X=0
  10 LET A=20
  20 LET T=10
  40 PRINT AT A,T,A$
  50 IF T<15 THEN LET T=T+INT (R
ND+2)
  70 IF T>5 THEN LET T=T-INT (RN
D+2)
  72 PRINT AT U,P,"+"
  75 SCROLL
  80 IF INKEY$="8" THEN LET P=P+
200/S
  90 IF INKEY$="5" THEN LET P=P-
1
 100 PRINT AT U,P,
 105 LET S=S+30-P
 110 LET X=X+P-T
 115 IF PEEK (PEEK 16398+256*PEE
K 16399)=CODE A$ THEN GOTO 1000
 117 PRINT "£"
 120 GOTO 40
1000 PRINT AT U,P,"£++++",INT (S
/X)
```

Posta

Inizia con questo numero la rubrica della posta. Abbiamo ricevuto molte lettere: non moltissime, ma abbastanza per impegnarci diversi giorni nella lettura (non solo dei testi, ma spesso anche dei programmi contenuti) e per "studiare" una risposta ai quesiti tecnici. Qui rispondiamo alle lettere di carattere generale. Nelle varie rubriche rispondiamo ai quesiti relativi appunto alle rubriche.

Continuate a scriverci. Il nostro indirizzo è

PERSONAL SOFTWARE
Via Rosellini 12
20124 MILANO

Tanti auguri

Spett. redazione,

dopo aver letto con piacere il primo numero di *Personal Software*, ho deciso di scrivervi per farvi i miei complimenti. A mio parere la vostra iniziativa desterà senz'altro l'attenzione che si merita, mancando finora in Italia un mezzo che svolga il compito di "educazione al software". L'argomento, infatti, è importantissimo per coloro che, dopo aver acquistato un personal computer, sentono la necessità di metterci dentro qualcosa di proprio, di originale, di diverso insomma dal programma che chiunque può acquistare. In altre parole, *Personal Software* mi sembra la rivista adatta per chi, invece di comprare, desideri creare.

Di conseguenza ho apprezzato molto la scelta del materiale presentato, che è stato attinto dalle più autorevoli fonti d'oltreoceano nel campo dell'"educazione al software", ed apprezzo ancor più decisamente l'impostazione della rivista. Trovo infatti che pubblicando articoli del tipo "come si fa" insieme ad altri di tipo "io ho fatto così", avete realizzato la raccomandazione di un saggio proverbio (si dice sia cinese) che dice pressapoco: "se regali un pesce ad un uomo che ha fame, lo sfamerai per un giorno; se gli insegni a pescare l'avrai sfamato per tutta la vita".

Spero perciò che la vostra rivista serva a favorire, qui da noi, la comparsa di programmatori creativi come è già successo in America, dove da anni alcune testate educano al software le fertili menti dei lettori.

Augurandovi una felice prosecuzione su questa strada, che in Italia è ancora tutta da percorrere, vi faccio una sola raccomandazione: quella di non diminuire il livello della sezione "articoli". Questo succederebbe pubblicando assieme al materiale tradotto qualcosa che, pur essendo meritevole perché italiano, risulti poi deplorevole perché "all'italiana".

Marco Morocutti
EC Elettronica, Brescia

Ringraziamo dei complimenti. Le sue parole esprimono esattamente le nostre convinzioni nel fare questa rivista.

I trucchi e le utility

Caro Direttore,

ho letto con molto piacere la sua nuova rivista; era ora che qualcuno si decidesse a compiere questo passo.

Sono un hobbysta in possesso di TRS-80 mod. 1 – 16 K cui, dopo un anno (e molti ripensamenti...) ho aggiunto una stampante, ed ora sto pensando ad un floppy...

Trascorro con il computer parte del mio tempo libero e cerco di farci un po' di tutto (al momento sostituisce la macchina da scrivere, ma senza word processor è un po' dura...).

Inoltre comincio a documentarmi e tentare qualche esperimento con interfacce ed attuatori con il "mondo esterno".

Ritengo che molti, come me, siano ormai ben stufi di leggere di prezzi, di beghe tra importatori e della macchina appena uscita migliore di tutte (anche di quella del giorno precedente).

Abbiamo voglia di farle camminare queste macchine e, se ci si guarda un po' in giro, avventurieri a parte, non si sente parlare d'altro che di contabilità in tutte le salse, paghe, eccetera. Un uso un po' troppo limitato, anche se mi rendo conto che il mercato ha le sue esigenze.

Trovo ottimo e ben assortito il contenuto del primo numero e per il futuro spero di trovare tanti "trucchi" ed utility.

Massimo Politi
Pescara

È stufo di leggere di prezzi, beghe e nuove macchine? Anche queste cose contribuiscono a fare il "mondo dei personal".

È vero che, per chi desidera solo divertirsi ed imparare con la propria macchina, sono più interessati i "trucchi e le utility". Ma il panorama editoriale dell'informatica è bello perché è vario.

La nostra rivista non è migliore delle altre: è diversa.

Spazio per il VIC

Caro Personal Software,

Finalmente sei arrivato. Penso che un po' tutti ti stessimo aspettando. Io sono uno di quei poveri disgraziati che hanno comperato il Commodore VIC 20 e si ritrovano senza poterlo usare nei modi dovuti in quanto non si trovano istruzioni esplicative su questa macchina. Ho comperato tutti i manuali che ho trovato sulla mia strada ma purtroppo solo uno è in italiano, sì, proprio quello pubblicato dalla Jackson, la quale ha tra l'altro annunciato un secondo manuale che ancora non si è visto. Oltre a questo ho il *VIC 20 Programmers Reference Guide*, pagato a peso d'oro, il quale in definitiva è appena superiore al "Jackson".

Infine ho il *VIC Revealed* che è il migliore, ma purtroppo in inglese, e io lo mastico poco, inoltre è pieno di errori di stampa, ed alcuni programmi non girano.

Vedendo gli annunci sulle varie rivi-

ste di parecchi possessori di VIC disperati, ho scritto pure io e vi chiedo a nome di tutti i Vicpatiti di dare uno spazio anche a questo micro, con notizie più comprensibili di quelle pubblicate su Bit riguardo ai cambiamenti di indirizzi, nonché informazioni sui manuali esistenti e sulle ditte italiane od estere che costruiscono periferiche per il VIC.

Vittorio Godio
Cesenatico

Per quanto riguarda il software, la nostra rivista sta pubblicando programmi adatti al VIC e lo farà anche in seguito. Abbiamo in preparazione degli articoli sulle caratteristiche software del VIC, e sappiamo che qualche casa editrice ha in programma qualcosa su questa macchina. Daremo tempestivamente tutte le informazioni di cui verremo in possesso. Intanto, in questo numero, gli appassionati di VIC diano un'occhiata alla nuova rubrica "I segreti dei personal".

Per il Sinclair

Gentile Redazione,
ho acquistato con molta curiosità il primo numero della Vostra rivista, perché sono un principiante nel campo dell'informatica e posseggo da un paio di mesi un Sinclair ZX 80 con la nuova ROM.

Ho quindi provato subito i due programmi da Voi riportati per lo ZX 80: "Odissea nello spazio" e "Roulette russa". Tengo a precisare che li ho caricati tenendo presenti le opportune differenze dovute alla nuova ROM, come ad esempio per la funzione RND, ma purtroppo entrambi i programmi erano troppo lunghi e non ci sono stati in memoria nemmeno sopprimendo tutte le istruzioni REM.

Desidererei perciò che Voi mi spiegaste gentilmente, a cosa è dovuto questo fatto. Colgo l'occasione per rivolgervi una seconda domanda: come posso fare per scrivere l'apostrofo col mio ZX 80 con nuova ROM?

Flavio Leoni
Milano

Lo ZX 81 (o ZX 80 con nuova ROM) occupa più byte in memoria sia per le istruzioni che per i dati, e quindi un programma di 1 K per lo ZX 80 non entra nello ZX 81 da 1 K. Le consigliamo di acquistare la RAM d'espansione.

Per quanto riguarda l'apostrofo, non esiste sulla tastiera, e quindi va sostituito con qualche altro carattere, oppure va evitato.

Per l'Olivetti

Da alcuni mesi mi interesso di personal computer e pertanto acquisto tutte le riviste del settore; più in là restringerò il campo a quelle che maggiormente mi soddisferanno...

Ho letto (e sto rileggendo) il n° 1 della Vs rivista (benedetti Voi che spillate le pagine anziché incollarle lungo il dorso!) e sono entusiasta del suo contenuto. Pur avendo fatto la mia scelta nello M20 della Olivetti (non ancora in mio possesso) ho trovato numerose e preziose notizie, leggendo ogni Vostro articolo.

A quando un "servizio" sullo M20?

Pietro Gentile
Napoli

Siamo anche noi alla ricerca di buoni articoli e programmi per l'M20. In questo momento non vi è abbondanza in materia perché il numero di M20 non è ancora quello dei PET ed Apple. Continui a leggere la nostra rivista: prima o poi troverà qualcosa di interessante per il suo computer, ma fin d'ora può trovare notizie, suggerimenti e tecniche generali che le sarà facile applicare all'M20.

Vuoi ordinare dei libri? Spedisci questo tagliando a: Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano.

LIBRI PERSONAL SOFTWARE

Nome Cognome

Indirizzo

Cap. Città

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

□ Pagherò al postino l'importo di L. .... + L. 1.500 per contributo fisso spese di spedizione

□ Allego assegno n° .... di L. .... (in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

□ Non abbonato □ Abbonato Data........................ Firma ....

N.B. È possibile effettuare versamenti anche sul ccp n° 11666203 intestato a: Gruppo Editoriale Jackson - via Rosellini, 12 - 20124 Milano. In questo caso specificare nell'apposito spazio sul modulo di ccp la causale del versamento e non inviare questo tagliando.

Vuoi ordinare dei libri? Spedisci questo tagliando a: Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano.

LIBRI PERSONAL SOFTWARE

Nome Cognome

Indirizzo

Cap. Città

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

□ Pagherò al postino l'importo di L. .... + L. 1.500 per contributo fisso spese di spedizione

□ Allego assegno n° .... di L. .... (in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

□ Non abbonato □ Abbonato Data........................ Firma ....

N.B. È possibile effettuare versamenti anche sul ccp n° 11666203 intestato a: Gruppo Editoriale Jackson - via Rosellini, 12 - 20124 Milano. In questo caso specificare nell'apposito spazio sul modulo di ccp la causale del versamento e non inviare questo tagliando.

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 1 N. 3 DICEMBRE 1982 L. 3.500

DIRETTORE RESPONSABILE: Giampietro Zanga

DIRETTORE: Mauro Boscarol

HANNO COLLABORATO A QUESTO NUMERO:
P. Dell'orco, A. Giovannetti, G. Williams, R. Baker, W. Douglas Maurer, J.S. Browning, C. Sintini, P. Canevarolo, N. Tonolli, G. Staluppi, Lexikon
Copertina: M. Elisa Rizzo
Grafica e Impaginazione: Fernanda Arzenton
Fotocomposizione: Composizioni Grafiche - Padova
Traduzioni: F. Santini, R. Pevarolo

CONTABILITA': Franco Mancini, Roberto Ostelli, Mariella Luciano, Franca Anelli, Sandra Cicuta, Gabriella Napoli

DIFFUSIONE E ABBONAMENTI: Luigi De Cao, Adela Bel Lozano, Ombretta Giannetto

AUTORIZZAZIONE ALLA PUBBLICAZIONE: Tribunale di Milano n. 69 del 20/2/1982

PUBBLICITA': Concessionario per l'Italia e l'Estero Reina s.r.l. Via Washington, 50 - 20146 Milano
Tel. (02) 4988066/7/8/9/060 (5 linee r.a.)
Telex 316213 REINA I

STAMPA: Arti Grafiche "La Cittadella" S.p.a. Pieve del Cairo (PV)

Concessionario esclusivo per la DIFFUSIONE in Italia e all'Estero:
SODIP - Via Zuretti, 25 - 20125 Milano

Spedizione in abbonamento Postale Gruppo III/70
Prezzo della rivista L. 3.500. Numero arretrato L. 6.000.
Abbonamento annuo (10 numeri) L. 28.000: per l'Estero L. 42.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano - mediante emissione di assegno bancario, cartolina vaglia o utilizzando il c/c Postale numero 11666203.
Per i cambi di indirizzo, indicare, oltre naturalmente al nuovo, anche l'indirizzo precedente, ed allegare alla comunicazione l'importo di L. 500, anche in francobolli.

GRUPPO EDITORIALE JACKSON Srl

DIREZIONE, REDAZIONE, AMMINISTRAZIONE:
Via Rosellini, 12 - 20124 Milano - Telefoni: 68.03.68 - 68.00.54

SEDE LEGALE: Via Vincenzo Monti, 15 - 20123 Milano

REDAZIONE USA: GEJ Publishing Group Inc. - 811 Haverhill Drive - 90407 Sunnyvale CA - Tel. (408) 7730103
DIREZIONE EDITORIALE: Giampietro Zanga e Paolo Reina
COORDINAMENTO EDITORIALE: Daniele Comboni

TUTTI I PREZZI E LE NOVITÀ

ANTEPRIMA "TRON"

L'ULTIMO DISNEY

I TRUCCHI DEI CAMPIONI

GIOCHIAMO CON I COMPUTER

IL PRIMO NUMERO È IN EDICOLA!